PlxMon98 User's Manual

Release 2.1, initial publishing November 23, 1998.

Document number: PCISDK-MON-210.doc

Table of Contents

1. Introduction

The PlxMon98 program is a simple and powerful tool for working with PLX devices. By making this product both easy to use and effective the user will be able to realize all the features of a given RDK. This is accomplished by grouping common registers into windows with a common theme, as well as providing useful utilities to manipulate the registers.

These dialog windows contain detailed information about each register and if necessary about the individual bits within. Combining this information with the programming algorithms included for accessing the various IOP components, and the ability to communicate over both PCI and Serial buses, PlxMon98 gives the tools needed to properly access any PLX RDK.

1.1 About This Manual

This manual is divided into 3 sections. After this introduction, a brief software tour is conducted. This will allow first time users to quickly set up PlxMon98 and see how some of the more common functions are used. The section following the tour is provided as a reference guide. In this area every feature found in PlxMon98 is defined alphabetically. The final section provides rapid data, such as a glossary of terms, a troubleshooting guide, and customer support contact information.

1.2 Conventions and Support

References to Windows NT assume Windows NT 4.0 or higher and will be shown as WinNT. Similarly, references to Windows 98 will be shown as Win98.

All references to IOP (I/O Platform) throughout this manual refer to the embedded hardware and all references to IOP software refer to the embedded software.

All values used in the manual are hexadecimal numbers, with the exception for memory sizes (used in Local Configuration Register screen). The prefix ‘0x’ has been omitted from all hexadecimal numbers and is not required when entering values for PLXMon98 fields.

It is important to note that PlxMon98 can only operate with a PLX device. It has been designed to work with the following PLX RDKs.

- PCI 9080RDK-401B;
- PCI 9054RDK-860;
- Any device that uses the PCI 9080 or PCI 9054 IC’s.

PLXMon98 has been tested to ensure compatibility with both Windows NT and Windows 98.

Note: *PLXMon98 is designed to run best with a high-resolution display, such as 1024x768 with 256 colors or better. Users choosing to run the software at lower resolutions will find it visually undesirable.*

1.3 PlxMon98 Feature List

PlxMon98 provides the following features:

- GUI screens that are based on PLX device registers;

- Compatible with the PCI 9080 and PCI 9054 devices;
- Split Screen Interface, allowing command line input while receiving serial data.
- Serial communications with an IOP's debug port. This feature is compatible with PLX's Back-End Monitor Level 1 protocol;
- A built-in downloader providing support for the following image standards: Motorola S-Record, IBM-401B Image Files (translated from ELF), COFF, and Binary. This feature supports downloading to RAM and FLASH devices. Downloading to the IOP can be done through either the PCI bus or Serial bus.
- PLX EEPROM Configuration screens to modify the contents of NM93CS46, NMCS56, and NMCS66 EEPROMs;
- Customizable Hot-Links. This feature allows users to launch Win32 compatible programs such as testing and sample programs;

***Note**: PlxMon98 is only compatible with PLX devices.*

1.4 Customer Support Information

Prior to contacting customer support, please ensure you have the following information:

1. You are situated close to the computer that has the PCI SDK installed;
2. Serial Numbers of the PLX PCI RDKs (if there is any in use with the PCI SDK);
3. Type of processor on the PLX PCI RDK;
4. Operating System version and type; and
5. Description of problem.

You may contact PLX customer support at:

Address:	PLX Technology, Inc. 390 Potrero Avenue Sunnyvale, CA 94086
Phone:	408-774-9060
Fax:	408-774-2169
Web:	http://www.plxtech.com

You may send email to one of the following addresses:

```
west-apps@plxtech.com
mid-apps@plxtech.com
east-apps@plxtech.com
euro-apps@plxtech.com
asia-apps@plxtech.com
```

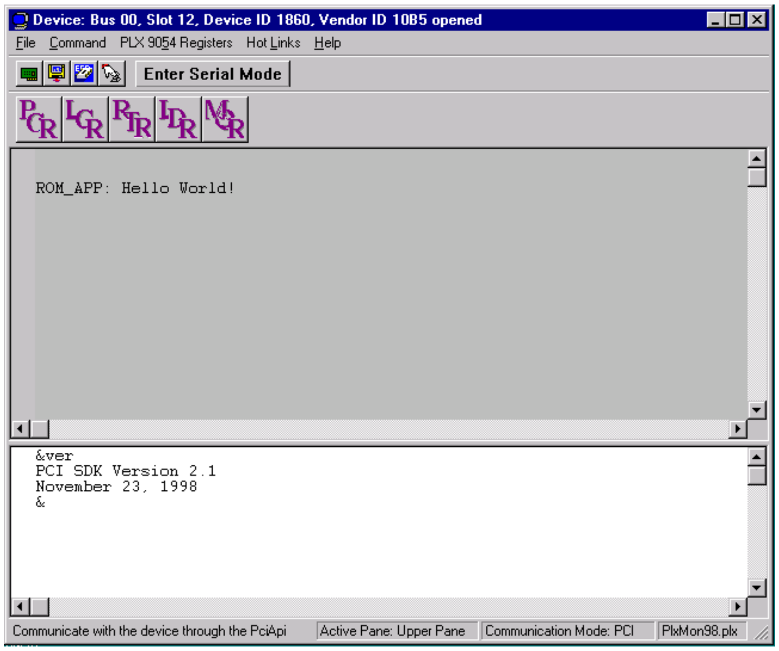
2. PlxMon98-A Brief Tour

This section will give a 10-minute tour of the PlxMon98 program. During this tour you will become familiar with how to setup the program and some of the more common features found in PlxMon98.

2.1 Starting and Configuration

PLXMon98 can be started by:

- Typing `PLXMon98` at the command prompt and pressing Enter; or
- Clicking on the icon in the PCI SDK folder in the Start Menu.


Device: Bus 00, Slot 12, Device ID 1860, Vendor ID 10B5 opened
File Command PLX 9054 Registers Hot Links Help
Enter Serial Mode
PCR LCR RTR IDR MCR

```
ROM_APP: Hello World!
```

```
&ver
PCI SDK Version 2.1
November 23, 1998
&
```

Communicate with the device through the PciApi | Active Pane: Upper Pane | Communication Mode: PCI | PlxMon98.plx

Figure 2-1 The PlxMon98 Interface

Depending on whether a supported PLX device is present, the monitor will enter PCI Mode or Serial Mode. If a PLX device is in a PCI slot on your computer you will be in PCI mode, otherwise you will enter Serial mode. It is possible to switch back and forth between the two modes after the program has started. In Figure 2-1 the greyed-out upper pane would be enabled in Serial Mode. Also note this program can be run on a second computer and used to do remote debugging through a serial port.

If PlxMon98 does not detect a standard PLX RDK on the system, it will notify the user that one does not exist. It is then necessary to add a device to the properties menu or verify that the

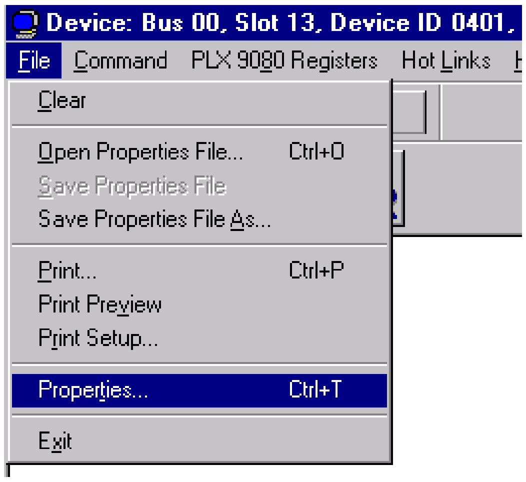
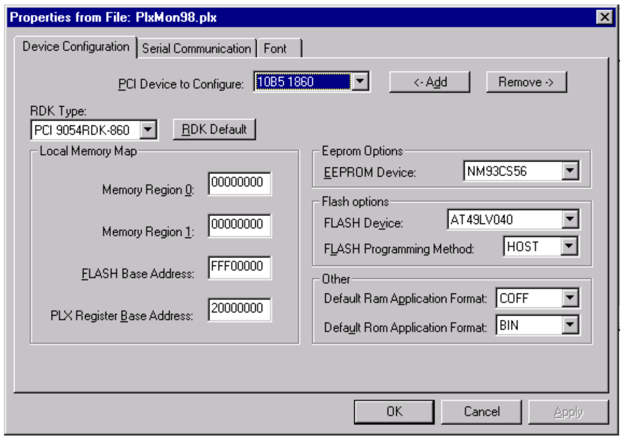

Device: Bus 00, Slot 13, Device ID 0401,
File Command PLX 9080 Registers Hot Links
Clear
Open Properties File... Ctrl+O
Save Properties File
Save Properties File As...
Print... Ctrl+P
Print Preview
Print Setup...
Properties... Ctrl+T
Exit

Figure 2-2 Selecting the Properties

device data is correct in the properties menu. Until properties are assigned to the specific Vendor and Device ID of your RDK, the program will not know the data needed to access the board. To edit these values select the

Properties item under the File pull-down menu. (You can also use the hot-key Ctrl-t).

The Properties menu first shows the Device Configuration page (see Figure 2-3). Most of this data will be correct if you are using a **standard PLX RDK**. These are the steps that should be followed however, to ensure proper operation of PlxMon98.


Properties from File: PlxMon98.plx

Device Configuration | Serial Communication | Font

PCI Device to Configure: 10B5 1860 | <- Add | Remove ->

RDK Type: PCI 9054RDK-860 | RDK Default

Local Memory Map
- Memory Region 0: 00000000
- Memory Region 1: 00000000
- FLASH Base Address: FFF00000
- PLX Register Base Address: 20000000

Eeprom Options
- EEPROM Device: NM93CS56

Flash options
- FLASH Device: AT49LV040
- FLASH Programming Method: HOST

Other
- Default Ram Application Format: COFF
- Default Rom Application Format: BIN

OK | Cancel | Apply

Figure 2-3 Device Configuration

1. If the Vendor and Device ID is not listed under the "PCI Device to Configure" combo box, then click the "ADD" push button to create

a new custom property entry for this device.
2. Now enter all the valid device information. By first clicking the `RDK Default` button, the default values for the RDK type selected will be entered on the screen. You should still verify that the Configuration EEPROM and FLASH EEPROM are the correct models.
3. Change the memory map, or set default datafile extension types, if necessary.

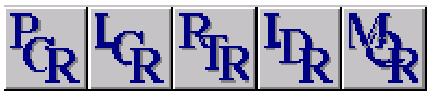
Now click the Serial Communications tab.

If you wish to use the debug port on your RDK, then you need to select the correct COM port. Also note if you are in Serial mode when doing this change, you must switch out then back into Serial mode for your changes to take effect. The IOP software on the PLX RDKs have been designed to accept 38400 Baud only. You can also change the timeout length if desired.

To see command line data in a different font, select the font tab to select many different styles and point sizes. Select the underline option and it will turn the display screen into lined pages.

2.2 Displaying Registers

Whether you are in PCI mode or Serial mode, you can access the registers on an RDK. It's as easy as clicking a button. The large buttons on the lower tier display various register sets. Click on the PCI Configuration Registers (PCR) button. A formatted PCI register set appears with some bit decoding already done. Greyed-out boxes indicate read-only windows. Now close this window and open the Local Configuration Register (LCR) window.



By clicking on an edit box, you can change the hexadecimal value then either close the window or move the cursor to another edit box to enter the value. Clicking on a check box will automatically update the register. Try clicking on the Details of any register box. The next dialog that appears will have check boxes to represent various bits of the register.

All the register capabilities mentioned above are applicable in Serial Mode as well as in PCI mode. PlxMon98 uses the interface provided by the Back End Monitor (BEM) to read and write to register locations. For more information on BEM, see either the PCI SDK Programmers Manual or the glossary.

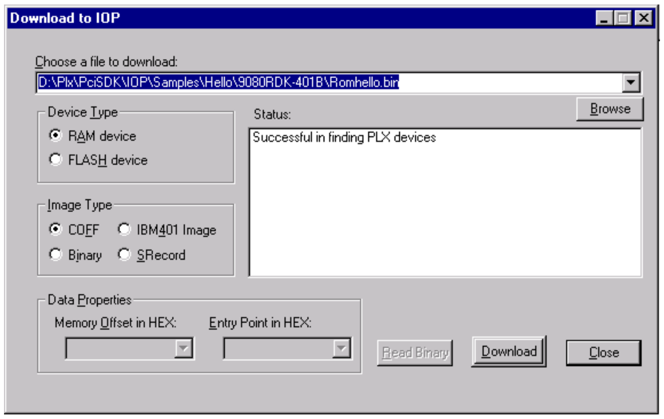
2.3 Popular Features

Now you can try using some tools that make PlxMon98 very useful. This section of the tour will give a brief overview of the Embedded loader as well as interactive DMA.

2.3.1 Downloading to the IOP

If you want to download an application to the IOP side, this is the feature you will need to use. Some of the features of the download utility include:

- Translation from different file formats including COFF, IBM Image file for the 401B, Motorola SRecord, and pure Binary.
- The ability to download the file to either RAM or FLASH ROM.
- Binary reads from FLASH ROM to a binary data file.
- Supports Serial downloads to RAM.



Choose a file to download: D:\Plx\PciSDK\IOP\Samples\Hello\9080RDK-401B\Romhello.bin

Device Type: [✓ RAM device] FLASH device

Status: Successful in finding PLX devices

Image Type: [✓ COFF] IBM401 Image Binary SRecord

Data Properties: Memory Offset in HEX: Entry Point in HEX:

Figure 2-4 Download To IOP

To familiarize yourself with downloading to an RDK, this tour will take you through the steps of downloading a RAM Hello Sample, and the Direct Master ROM sample, both of which are found in the PCI SDK 2.1. The examples below demonstrate downloading using the PCI 9054RDK-860 but the methods also apply to the PCI 9080RDK-401B.

Downloading to RAM

1. You must first select either a Serial or PCI channel before opening the Download window. If performing a PCI download you can connect a serial port to another running PlxMon98 application, and see the download in progress.
2. Open the Download Window by clicking the download to embedded icon, which looks like a small disk with a downward facing arrow.
3. In the Device Configuration information, RAM data is stored in COFF format for the currently selected device. Therefore the default file type will be COFF.

4. Go to the directory: <Install-Path>\Plx\PciSdk\IOP\Samples\Hello\9054RDK-860 and select the file RamHello.cof.
5. Click on download. You can verify that the program was successfully downloaded by reading the status screen (See Figure 2-4).

Burning a FLASH ROM

Note: *If in the course of writing a program to the FLASH an error occurs, do not shut down the computer until you have re-burned a valid ROM program. Failure to do so will require removal of the FLASH ROM chip and re-burning in an external device programmer.*

1. Downloads to FLASH are currently only supported via the PCI bus. The option to FLASH program will be disabled if entered in Serial mode. Select the FLASH device.
2. After selecting FLASH, set the desired image type. It will also be necessary to give an offset from the physical FLASH address set in the configuration menu. This is necessary because certain RDKs produce FLASH data to be downloaded to different addresses. The following chart describes the offsets used for each RDK.

RDK Type	FLASH OFFSET (in HEX)
PCI 9054RDK-860	0
PCI 9080RDK-401B	60000

Valid FLASH Offsets

3. Go to the directory: <Install-Path>\Plx\PciSdk\IOP\Samples\Dmaster\9054RDK-860 and select the file RomDM.bin.
4. Now click on download. Again you can verify that the download was successful by reading from the serial port on another PlxMon98, and by reading the Status window as well. A good way to test the ROM code is to reset the RDK, and verify the same program is run after the reset.

For a more complete description of the IOP download function consult the reference section.

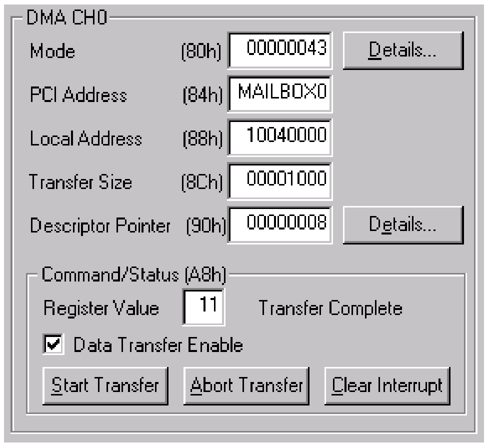
2.3.2 DMA Transfers

DMA transfers can be used to copy data rapidly between the IOP and the PCI bus. This PCI bus could be a user-mapped common buffer, or another RDK's local space window. This example will demonstrate how to set up a simple IOP to PCI transfer.

1. To set up the DMA transfer, some registers must be properly initialized. First click on the LDR button, to display the registers to be modified.

2. Set up the DMA transfer as seen in Figure 2-4. The minimum requirement for set up of the mode register is to set the Ready Input Enable. It is also recommended, but not required to set the bus width to 32 bit. These combinations give the mode register a value of 43. The value in PCI Address, listed here as "MAILBOX 0" is any valid physical PCI address. The SDK driver will allocate a fixed size buffer (10000 bytes by default) and place a pointer to this buffer in mailbox 0. The Local Address given here is valid for both supported RDKs up to a size of 40000 bytes (hex). Transfer direction is set within the Descriptor Pointer, the value 8 indicates an IOP to PCI DMA transfer.

Note: *Channel1 DMA registers and threshold data registers are not displayed here and can be ignored for this example.*



DMA CH0

Mode	(80h)	00000043	Details...
PCI Address	(84h)	MAILBOX0	
Local Address	(88h)	10040000	
Transfer Size	(8Ch)	00001000	
Descriptor Pointer	(90h)	00000008	Details...

Command/Status (A8h)

Register Value 11 Transfer Complete

[✓] Data Transfer Enable

Figure 2-5 DMA Channel 0 registers

3. To start the transfer, first enable the transfer by clicking the `Data Transfer Enable` bit. Clicking on `Start Transfer` will begin the operation. As the transfer size is so small, The DMA status will only flicker to `Transfer in Progress` before returning to `Transfer Complete`.

4. To verify the data was retrieved, read the system variable `hbuf` which is the user pointer to the PCI common buffer.

Note: *Dereferencing the PCI buffer addresses (using the dl command, for example) in user space will more than likely produce undesired effects.*

This concludes the tour of PlxMon98. Remember, you had any problems while following this tutorial, check the troubleshooting section of the manual or call technical support. Information on both these options are found in the appendices.

3. PlxMon98-Reference

The reference section is provided to give detailed information about any one feature of the PlxMon98 application. It is organized alphabetically by feature title.

3.1 Access Mode

The two types of access to the RDKs are via the PCI bus and via the Serial port. See Figure 3-1.

3.1.1 PCI Mode

When PCI mode is selected all communication between PLXMon98 and the PLX chip is done via the PCI SDK API and device driver. This method will be familiar to users of PLXMon97 and PlxMon98 v2.0. The mode is selected by toggling the PCI button on the toolbar between PCI and Serial. Essentially this results in all communication to the PLX device via its PCI Bus interface.

3.1.2 Serial Mode

When serial communication mode is selected all communication between PLXMon98 and the PLX device bypasses the PCI SDK API and driver. Instead, PLXMon98 communicates to the IOP's BEM module (refer to PCI SDK Programmer's Manual for BEM details). Essentially this results in all communication to the PLX device via its Local Bus interface. Commands are limited to local reads and writes, and performing a local reset.

The following PLXMon98 features are not available when operating in serial mode:

- Downloading to FLASH;
- EEPROM Configuration; and
- Selecting Devices.

Note: *To use this mode, your IOP software must contain the BEM module. All PLX RDKs support this mode by default.*

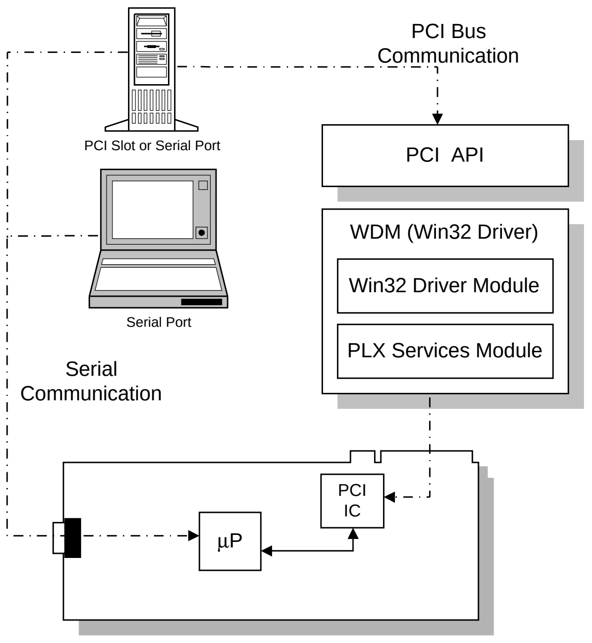
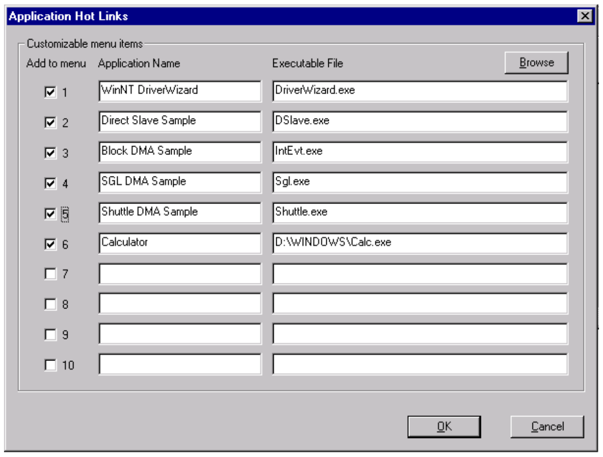


Figure 3-1 PCI vs. Serial Data Flow

3.2 Application Hot Links

PLXMon98 offers the capability for users to add hot links to popular SDK applications. Typical applications that users may like to add hot links to are SDK samples, manufacturing test software, or custom applications.


Application Hot Links

Customizable menu items

Browse

Add to menu	Application Name	Executable File
☑ 1	WinNT DriverWizard	DriverWizard.exe
☑ 2	Direct Slave Sample	DSlave.exe
☑ 3	Block DMA Sample	IntEvt.exe
☑ 4	SGL DMA Sample	Sgl.exe
☑ 5	Shuttle DMA Sample	Shuttle.exe
☑ 6	Calculator	D:\WINDOWS\Calc.exe
☐ 7		
☐ 8		
☐ 9		
☐ 10		

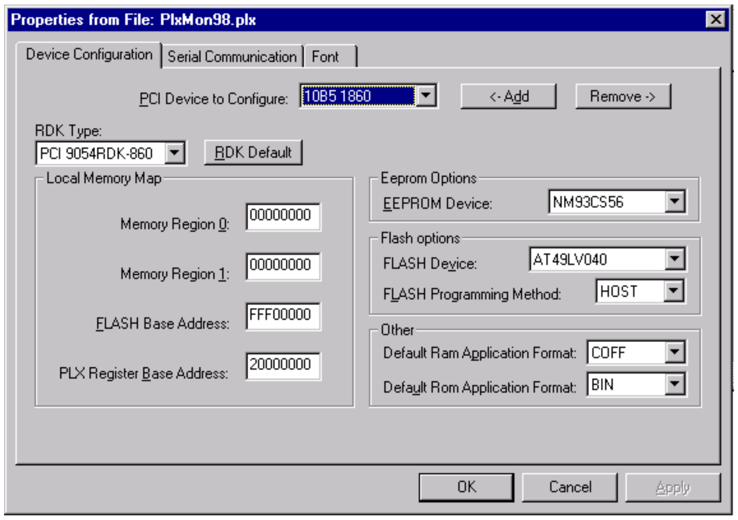
OK Cancel

Figure 3-2 Application Hot Links dialog box

To use this feature you need to enter the Application Name and associated path to the executable in the options menu. They will then appear in the Hot Link pull down menu.

3.3 Device Configuration

This dialog window can be found by pulling down the File menu then selecting Properties. PLXMon98 remembers program options for specific PLX RDKs and custom boards. These options are used throughout PLXMon98 to setup default values for various program options. For example the memory map is essential for all direct register accesses to the IOP.


Properties from File: PlxMon98.plx

Device Configuration | Serial Communication | Font

PCI Device to Configure: 10B5 1860   <- Add   Remove ->

RDK Type: PCI 9054RDK-860   RDK Default

Local Memory Map
- Memory Region 0: 00000000
- Memory Region 1: 00000000
- FLASH Base Address: FFF00000
- PLX Register Base Address: 20000000

Eeprom Options
- EEPROM Device: NM93CS56

Flash options
- FLASH Device: AT49LV040
- FLASH Programming Method: HOST

Other
- Default Ram Application Format: COFF
- Default Rom Application Format: BIN

OK Cancel Apply

Figure 3-3 Device Configuration dialog box

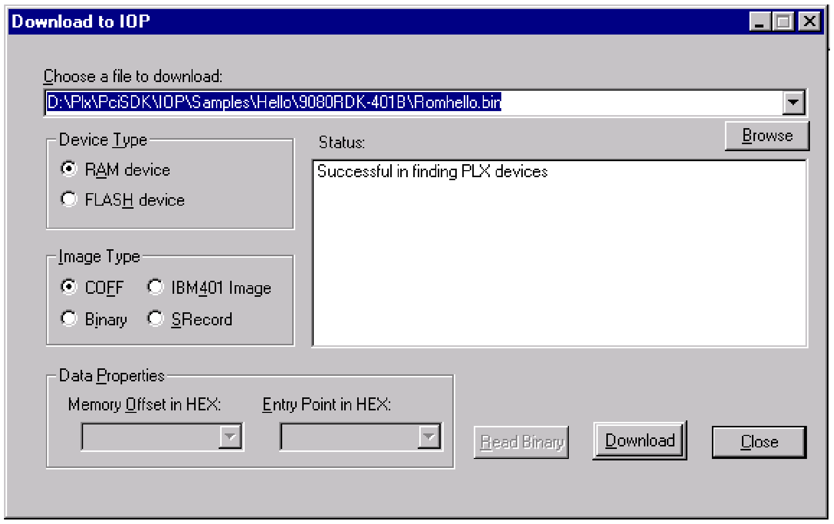
PlxMon98 uses the Vendor ID and Device ID to recognize a **supported** device. By adding a new entry to this list, a custom ID combination can be supported. Figure 3-3 shows the settings for the PLX 9080RDK-401B board. Similar settings must be selected for each RDK that PlxMon98 is used on.

When adding a new device, first enter the custom Vendor ID and Device ID. Then select it's `RDK type` (if applicable). On hitting `RDK Default`, the normal settings for that RDK will be entered. Press Apply or OK to enter the data into the Properties file.

3.4 Downloading IOP Applications

In order to download to and execute programs on the IOP, the `Download to Embedded` command is provided. This utility can be run by clicking on the icon, or by selecting it under the `Command` pull-down menu.

Files can be sent to either a RAM device or FLASH device and the base address is programmable.


Download to IOP

Choose a file to download: D:\Plx\PciSDK\IOP\Samples\Hello\9080RDK-401B\Romhello.bin

Device Type: (● RAM device) ○ FLASH device

Image Type: (● COFF) ○ IBM401 Image ○ Binary ○ SRecord

Status: Successful in finding PLX devices

Data Properties: Memory Offset in HEX: Entry Point in HEX:

Figure 3-4 File download dialog box

Typically, all default values that are already selected will be correct for the download you wish to do. By selecting either RAM or FLASH, the file format will automatically be changed to the format that is specified for this device in the Device Config. menu (Sect. 3.3). These features can be overridden, and by doing so the user should be knowledgeable about the format of these files and the memory map of the RDK being programmed.

Note: *Programming the FLASH can be dangerous and it is important that the setup parameters are correct before the download is attempted. Data like RDK Type, Flash Address, Programming Method, and Memory Offset should be known beforehand.*

To read the data on a FLASH device into a file, the `Read Binary` button can be used in conjunction with the `memory offset` window. This data will be retrieved unformatted and stored as a pure binary file.

This utility also has the ability to program a device through the serial debug port. Currently the serial download supports only RAM programming.

3.5 Font Configuration

This dialog screen can be reached from the Properties option in the File pull-down menu. The font and point size will be changed in the display screen only. The appearance of the data in the dialog boxes will not be changed.

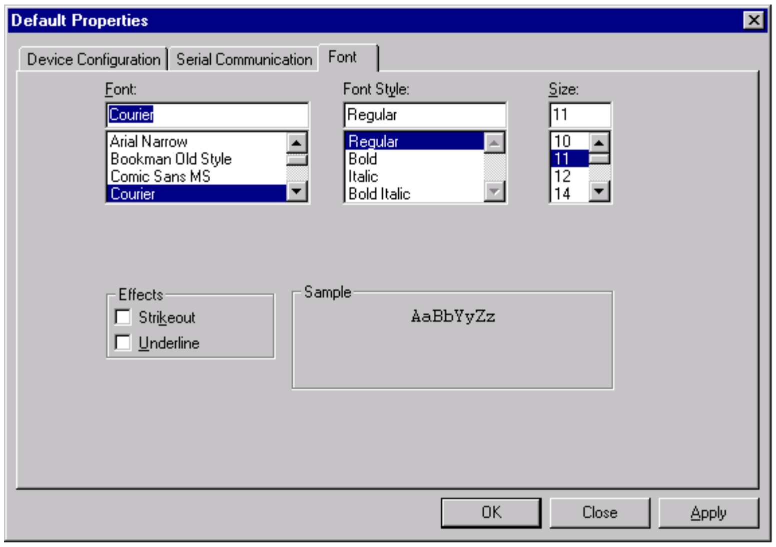
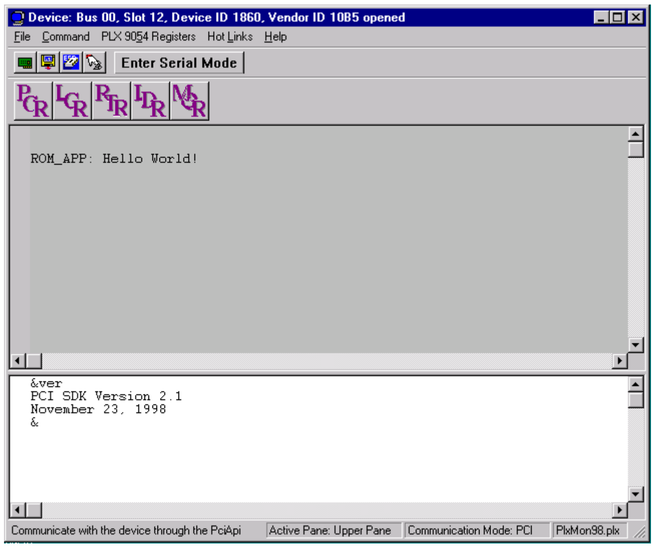


Figure 3-5 Font Select dialog box

3.6 The Interface

The PLXMon98's main interface, shown in Figure 3-6 contains:

- A drop-down menu bar, with the five main drop-down menus. Depending on which PLX RDK is selected, certain options will be available. The PCI 9054 menu is shown here;
- A split screen interface used for simultaneous Command Line Interface (Lower Pane) and Serial Access (Upper Pane). Use the F6 key to toggle between active panes.
- An optional toolbar (for a full view, see Section 3.7.1);
- The status bar which reports the configuration file being used;
- The Enter Serial Mode/Enter PCI Mode button that toggles communication mode between PCI and serial communications.


Device: Bus 00, Slot 12, Device ID 1860, Vendor ID 10B5 opened

File Command PLX 9054 Registers Hot Links Help

Enter Serial Mode

```
ROM_APP: Hello World!
```

```
&ver
PCI SDK Version 2.1
November 23, 1998
&
```

Communicate with the device through the PciApi | Active Pane: Upper Pane | Communication Mode: PCI | PlxMon98.plx

Figure 3-6 The PlxMon98 Interface

3.6.1 The PLXMon98 Toolbar

The PLXMon98 toolbar, shown in Figure 3-7, serves as an optional shortcut to the drop-down menu commands.

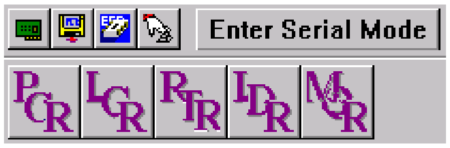


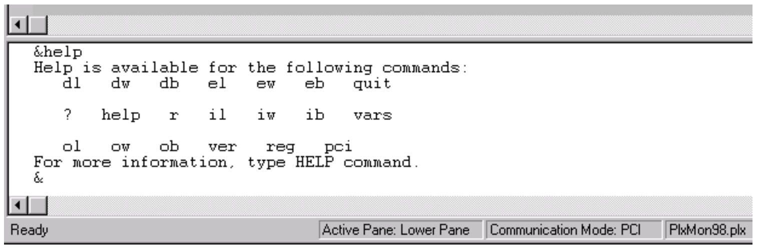
Figure 3-7 PlxMon98 Toolbar

3.6.2 Status Bar

The status bar provides simple and useful tips. When the mouse is pointing on an object or a button, in the main window of PLXMon98 the status bar displays some information on it. Tool-tips are also displayed when mouse pointer is held over an object for a short period.

3.6.3 Command Line Interface (CLI)

At the & prompt which is located in the Lower Pane, various commands can be entered to get information on the RDK that is selected. This command line can be used in both PCI and Serial modes. To get a list of valid commands at any time in PlxMon98, at the command line, type `help` or `?.` The following sections will describe all the valid commands.



```
&help
Help is available for the following commands:
   dl   dw   db   el   ew   eb   quit

   ?   help   r   il   iw   ib   vars

   ol   ow   ob   ver   reg   pci
For more information, type HELP command.
&
```

Figure 3-8 Valid Commands in PlxMon98

Note: *The CLI is not case sensitive.*

3.6.3.1 Displaying Memory via Memory Cycles, (dl, dw, db)

These commands display different sizes of data that are accessed through memory cycles. Using `dl` will return a 32-bit value, `dw` will return a 16-bit value, and `db` will return a 8-bit value.

They follow the format:

```
<command> <address> [[l] bytelength]
```

by default the bytelength is 80 [hex] bytes. Typing the command again with no arguments will make PlxMon98 continue to display the range with the same `bytelength` as before.

Note: *This command will dereference the address range given. If an invalid address range is supplied, unpredictable results may occur.*

3.6.3.2 Displaying Memory via I/O cycles, (il, iw, ib)

The iX commands are similar in syntax with the dX commands except they access memory using I/O cycles instead of memory cycles. They follow the format:

```
<command> <address> [[l] bytelength]
```

By default, the bytelength is dependant on the command. Using `il` will return 32 bits of data, `iw` will return 16 bits, and `ib` will return 1 byte. All these lengths can be overridden, however. By re-typing the command with no arguments, PlxMon98 will continue to display the memory locations using the size of the data retrieved as the increment size.

Note: *This command will dereference the address range given. If an invalid address range is supplied, unpredictable results may occur.*

3.6.3.3 Writing Memory via Memory Cycles, (el, ew, eb)

Again the syntax of the write using memory cycles is simlar to the dX commands. Using `el` will write values as 32 bit wide objects, `ew` will write on 16 bit values, and `eb` will write a byte at a time.

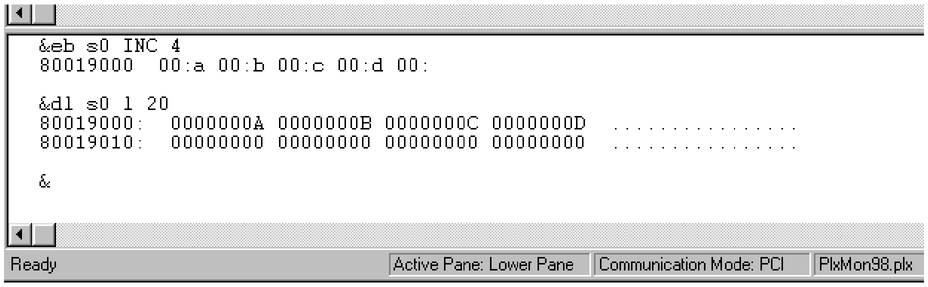
They follow the format:

```
<command> <address> [INC increment] [value]
```

While the input address is required, both the value and the increment parameter are optional. By not entering the [value] parameter, the program will query you for data in interactive mode. Interactive mode allows the user to press the space bar between after typing the value he/she want to enter and PlxMon98 will auto-increment the write address the size of the data being entered. Press the enter key to exit the command. By changing the `INC` parameter in the command line above, the auto-increment value can be changed. What follows is a brief example of how to use this command.

If the user types the command `eb s0 INC 4`, they wish to interactively write bytes to the location `s0` (which is a system label, more on this in section 3.7.3.6). Every time the user enters a value and hits space the program will query for the previous address plus INC which is 4. On display of the memory range the user should see the following screen.

The 00: bytes that appear before each user entry are the previous values that are about to be overwritten by the user's data.



```
&eb s0 INC 4
80019000  00:a 00:b 00:c 00:d 00:

&dl s0 l 20
80019000:  0000000A 0000000B 0000000C 0000000D  ................
80019010:  00000000 00000000 00000000 00000000  ................

&
```

Figure 3-9 Interactive Mode

Note: *This command will write user values to the address range given. If an invalid address range is supplied, unpredictable results may occur.*

3.6.3.4 Writing Memory via I/O cycles, (ol, ow, ob)

The oX commands are simpler syntactically than the memory cycle writes. Each command writes a different sized data object to a port address. They require only two parameters:

```
<command> <address> <value>
```

Note: *This command will write user values to the address range given. If an invalid address range is supplied, unpredictable results may occur.*

3.6.3.5 The pci command

This command allows read/write access the PCI configuration registers. It's syntax is as follows:

```
pci <pci offset> [value]
```

To write to the required offset, just add the value to write; otherwise the value will be displayed as a 32-bit register value.

Note: *the offset will be different depending on the Access mode: PCI or Serial.*

3.6.3.6 The quit command

The `quit` command terminates the application PlxMon98.

3.6.3.7 The reg command

The `reg` command allows users access to the PLX RDK's local register sets. Data can be read or written in 32 bit sizes at a given byte boundary. The syntax of the command is as follows:

```
reg <register offset> [value]
```

If a `[value]` is given, the command will write the data to the specified address.

3.6.3.8 The repeat command (r)

The repeat command can is used to make PlxMon98 repeat the command types before the `r` a set number of times. It's syntax is as follows:

```
[command] r [iterations]
```

If the number of `iterations` is not given, then PlxMon98 will execute the command indefinitely until the user hits a key. The command to be repeated must be in the same expression as the `r` command.

3.6.3.9 User Variables (vars)

PlxMon98 creates some user labels as a mnemonic aid for common memory locations. These strings can be used interchangeably with the values they represent.

This table lists the variables set by PlxMon98 and what memory ranges they represent:

Variable Name	Description	PCI or Serial
hbuf	User mapped region of the PCI common buffer.	PCI
PlxRegBase	Memory-mapped PLX RDK registers	PCI
PlxIoBase	I/O mapped PLX RDK registers	PCI
PlxLocalBusBase	Memory address which represents the Local Bus	PCI
s0	Local Space 0	PCI
s1	Local Space 1	PCI
s2	Local Space 2	PCI
s3	Local Space 3	PCI
PciVar	Local address base of PCI registers	Serial
LcrVar	Local address base of local configuration registers	Serial
RtrVar	Local address base of runtime registers	Serial
DmaVar	Local address base of DMA registers	Serial
MqrVar	Local address base of messaging queue registers	Serial

Figure 3-10 User Variables and their definitions

3.6.3.10 The ver command

Displays the version data contained in the PCI SDK software release. This version of PlxMon98 is compatible only with PCI SDK version 2.1.

3.7 Print, Print Preview, and Print Setup

The print commands can be found under the File pull-down menu. These selections enable the user to create a formatted picture of the display window. Print preview will allow you to see the formatted screen before you print.

3.8 Register Access/Register Sets

The contents of registers can be represented in one of two ways in PlxMon98. Usually when a full 32-bit register is being displayed, it is shown in an edit box in hexadecimal format *(the 0x prefix is implied)*. These boxes, if not greyed-out, can be modified by typing in new values. The value will be updated when the user closes the window or when the cursor is moved to another edit box.

Check boxes are also used to display and change individual bits on a register. If the check box is on a main dialog window then a state change is immediate. If the check box is in a Details dialog with other checkboxes, then only upon closing the dialog are the changes made.

Please refer to Appendices A and B for specific details on the PCI 9080 and PCI 9054 register set, respectively.

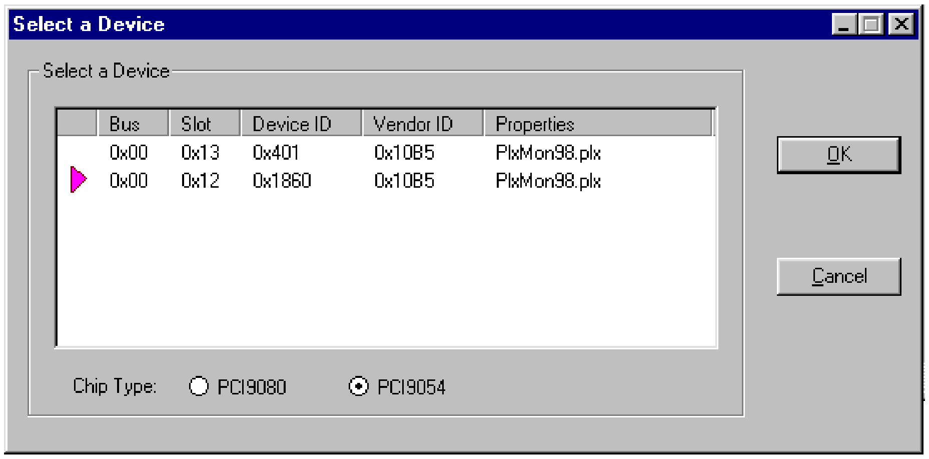
3.9 The Reset Button

Found on the taskbar, the reset button signals the PLX RDK that is currently selected to reset itself. This action can be taken during both PCI and Serial modes.

Note: *The method used to reset is customized for PLX RDKs. This feature should not be used on devices that do not support the reset algorithm. Consult the BSP source code for information on the algorithm used.*

3.10 Selecting Devices

The `Select A PCI Device` menu item, shown in Figure 3-11, is used to select a PCI device to access. The pointer points to the currently selected device.



Select a Device

	Bus	Slot	Device ID	Vendor ID	Properties
	0x00	0x13	0x401	0x10B5	PlxMon98.plx
▶	0x00	0x12	0x1860	0x10B5	PlxMon98.plx

Chip Type: ○ PCI9080 [✓ PCI9054]

Figure 3-11 Device Select dialog box

To select a new device, move the highlight to the desired PCI device by using either the mouse or the cursor arrow keys, and click the `OK` button or press the Enter key. The pointer will not move to the new selection until OK button is pressed or until the item is double-clicked.

The chip type radio buttons will indicate which PLX device is present on the selected device.

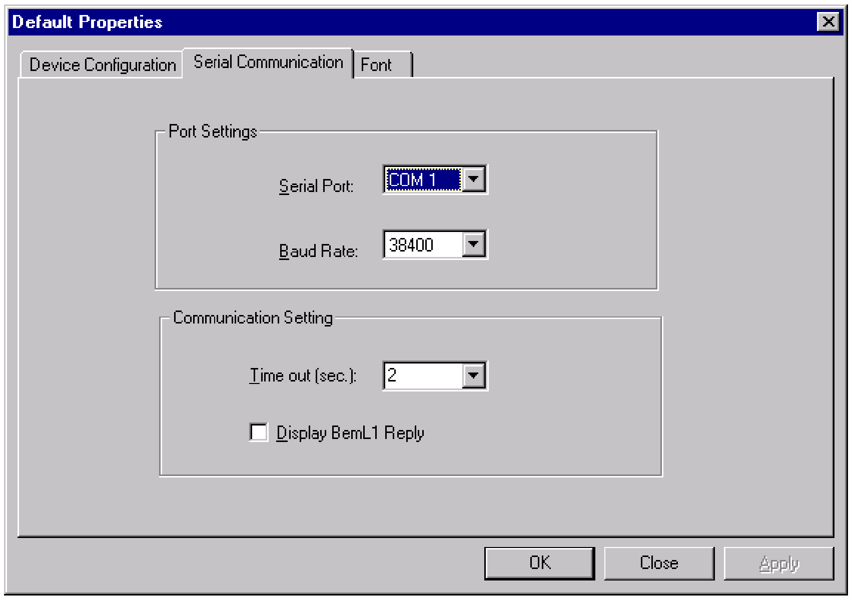
Note: *Only PLX devices are shown in the list.*

3.11 Serial Configuration

PLXMon98 offers the capability to communicate with the PLX device through the serial port. To do so you must configure the appropriate serial port settings as shown in the figure below.

Communication ports COM1 through COM4 are supported. Baud rates 9600, 19200, 38400, and 57600 are supported.

***Note:** All PLX RDK's should be configured to 38400 baud.*



Port Settings

Serial Port: COM 1

Baud Rate: 38400

Communication Setting

Time out (sec.): 2

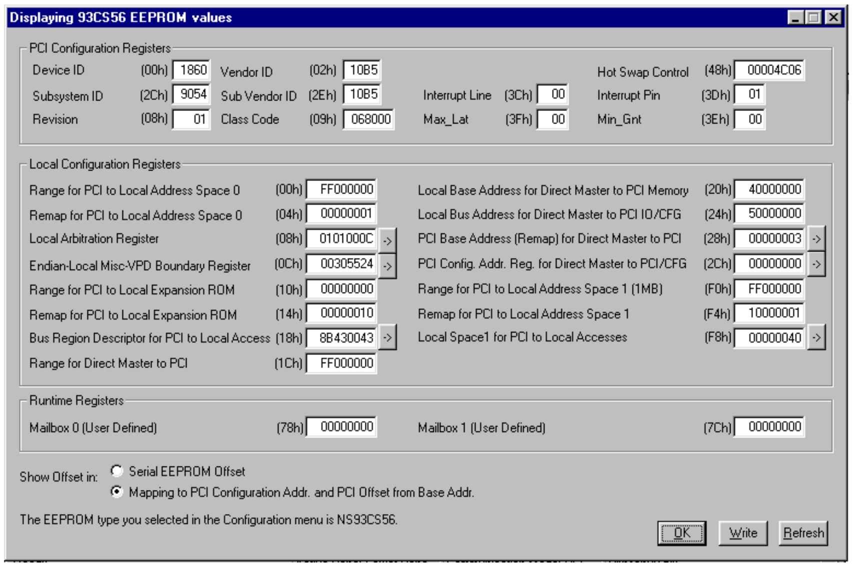
☐ Display BemL1 Reply

Figure 3-12 Serial Communications Properties dialog box

3.12 Serial EEPROM Access

Pressing the Serial EEPROM button on the toolbar of PlxMon98 will start the dialog box seen below. No data will be written until the `Write` button is pressed. Clicking `Refresh` will overwrite any changes made.

***Note:** Having the wrong Serial EEPROM type selected in the Device Config. Menu can cause PlxMon98 to read/write invalid data.*



PCI Configuration Registers

Device ID	(00h)	1860	Vendor ID	(02h)	10B5				Hot Swap Control	(48h)	00004C06
Subsystem ID	(2Ch)	9054	Sub Vendor ID	(2Eh)	10B5	Interrupt Line	(3Ch)	00	Interrupt Pin	(3Dh)	01
Revision	(08h)	01	Class Code	(09h)	068000	Max_Lat	(3Fh)	00	Min_Gnt	(3Eh)	00

Local Configuration Registers

Range for PCI to Local Address Space 0	(00h)	FF000000	Local Base Address for Direct Master to PCI Memory	(20h)	40000000
Remap for PCI to Local Address Space 0	(04h)	00000001	Local Bus Address for Direct Master to PCI IO/CFG	(24h)	50000000
Local Arbitration Register	(08h)	0101000C	PCI Base Address (Remap) for Direct Master to PCI	(28h)	00000003
Endian-Local Misc-VPD Boundary Register	(0Ch)	00305524	PCI Config. Addr. Reg. for Direct Master to PCI/CFG	(2Ch)	00000000
Range for PCI to Local Expansion ROM	(10h)	00000000	Range for PCI to Local Address Space 1 (1MB)	(F0h)	FF000000
Remap for PCI to Local Expansion ROM	(14h)	00000010	Remap for PCI to Local Address Space 1	(F4h)	10000001
Bus Region Descriptor for PCI to Local Access	(18h)	8B430043	Local Space1 for PCI to Local Accesses	(F8h)	00000040
Range for Direct Master to PCI	(1Ch)	FF000000			

Runtime Registers

Mailbox 0 (User Defined)	(78h)	00000000	Mailbox 1 (User Defined)	(7Ch)	00000000

Show Offset in: ○ Serial EEPROM Offset [✓ Mapping to PCI Configuration Addr. and PCI Offset from Base Addr.]

The EEPROM type you selected in the Configuration menu is NS93CS56.

Figure 3-13 Serial EEPROM dialog box

Appendix A. The PCI 9080 Register Set

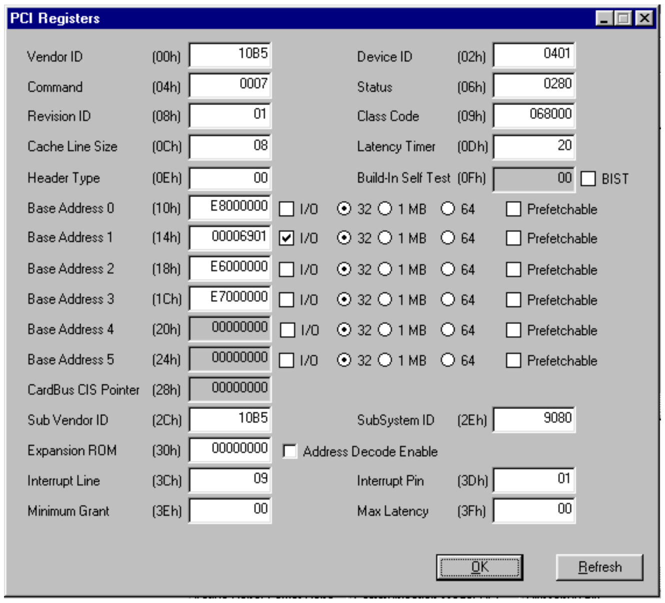
Each of the PCI 9080's register groups has a distinct dialog box. Each dialog box has the register values, the register's PCI based addresses, and a description of the register. Some registers have check boxes and radio buttons to help in describing and setting the register values. When required additional dialog boxes are available for more complex registers.

A.1 The Register Group Dialog Boxes

The PLXMon98's toolbar contains five buttons for register accesses. They are for `PCI Configuration Registers (PCR)`, `Local Configuration Registers (LCR)`, `Runtime Registers (RTR)`, `Local DMA Registers (LDR)`, and `Messaging Queue Registers (MQR)`.

A.1.1 PCI Configuration Register Group Dialog Box

The grayed text, in Figure A-1, on the `PCI Configuration Registers` dialog box indicates that the values cannot be modified using this dialog box. The radio buttons and check boxes, indicate the current settings of the register bit fields. To update the contents of the dialog box push the `Refresh` button.

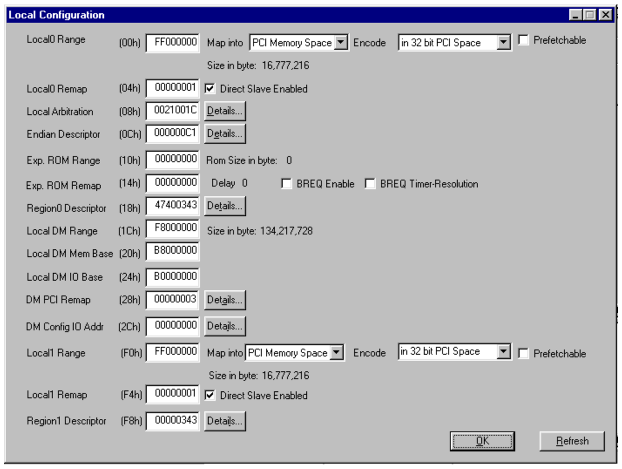


Vendor ID	(00h)	10B5		Device ID	(02h)	0401	
Command	(04h)	0007		Status	(06h)	0280	
Revision ID	(08h)	01		Class Code	(09h)	068000	
Cache Line Size	(0Ch)	08		Latency Timer	(0Dh)	20	
Header Type	(0Eh)	00		Build-In Self Test	(0Fh)	00	☐ BIST
Base Address 0	(10h)	E8000000	☐ I/O	[✓ 32] 1 MB 64			☐ Prefetchable
Base Address 1	(14h)	00006901	☑ I/O	[✓ 32] 1 MB 64			☐ Prefetchable
Base Address 2	(18h)	E6000000	☐ I/O	[✓ 32] 1 MB 64			☐ Prefetchable
Base Address 3	(1Ch)	E7000000	☐ I/O	[✓ 32] 1 MB 64			☐ Prefetchable
Base Address 4	(20h)	00000000	☐ I/O	[✓ 32] 1 MB 64			☐ Prefetchable
Base Address 5	(24h)	00000000	☐ I/O	[✓ 32] 1 MB 64			☐ Prefetchable
CardBus CIS Pointer	(28h)	00000000					
Sub Vendor ID	(2Ch)	10B5		SubSystem ID	(2Eh)	9080	
Expansion ROM	(30h)	00000000	☐ Address Decode Enable				
Interrupt Line	(3Ch)	09		Interrupt Pin	(3Dh)	01	
Minimum Grant	(3Eh)	00		Max Latency	(3Fh)	00	

Figure A-1 PCI Configuration Registers dialog box

A.1.2 Local Configuration Register Group Dialog Box

The Local Configuration register values are updated through the related edit and dialog boxes, in Figure A-2, respectively. The `size` text box reflects the value (in bytes) of the associated register. The memory size is calculated from corresponding register value and cannot be modified directly. To change the memory size, modify the associated register.



Local0 Range	(00h)	FF000000	Map into PCI Memory Space; Encode in 32 bit PCI Space; ☐ Prefetchable; Size in byte: 16,777,216
Local0 Remap	(04h)	00000001	☑ Direct Slave Enabled
Local Arbitration	(08h)	0021001C	Details...
Endian Descriptor	(0Ch)	000000C1	Details...
Exp. ROM Range	(10h)	00000000	Rom Size in byte: 0
Exp. ROM Remap	(14h)	00000000	Delay 0; ☐ BREQ Enable; ☐ BREQ Timer-Resolution
Region0 Descriptor	(18h)	47400343	Details...
Local DM Range	(1Ch)	F8000000	Size in byte: 134,217,728
Local DM Mem Base	(20h)	B8000000	
Local DM IO Base	(24h)	B0000000	
DM PCI Remap	(28h)	00000003	Details...
DM Config IO Addr	(2Ch)	00000000	Details...
Local1 Range	(F0h)	FF000000	Map into PCI Memory Space; Encode in 32 bit PCI Space; ☐ Prefetchable; Size in byte: 16,777,216
Local1 Remap	(F4h)	00000001	☑ Direct Slave Enabled
Region1 Descriptor	(F8h)	00000343	Details...

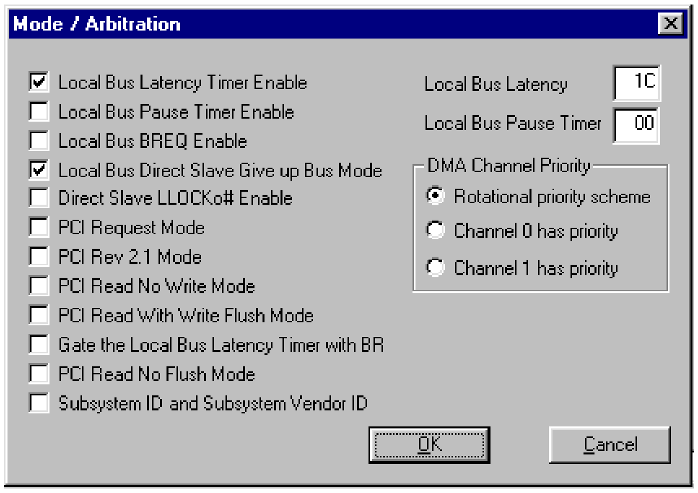
Figure A-2 Local Configuration Registers dialog box

Six registers within the Local Configuration Register Group have a more detailed dialog box and are as follows:

- The Mode/Arbitration dialog box;
- The Endian Descriptor dialog box;
- The Space 0/Exp ROM dialog box;
- The DM PCI Remap dialog box;
- The DM Config IO Address dialog box; and,
- The Space 1 dialog box.

The Mode/Arbitration Dialog Box

The Mode/Arbitration dialog box provides information on the current value Local/DMA Arbitration register and allows modification of that value (see Figure A-3).



Mode / Arbitration

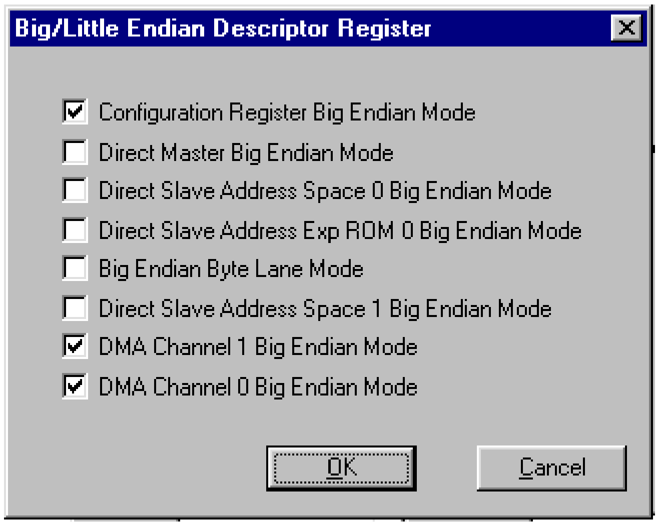
- [✓] Local Bus Latency Timer Enable
- [ ] Local Bus Pause Timer Enable
- [ ] Local Bus BREQ Enable
- [✓] Local Bus Direct Slave Give up Bus Mode
- [ ] Direct Slave LLOCKo# Enable
- [ ] PCI Request Mode
- [ ] PCI Rev 2.1 Mode
- [ ] PCI Read No Write Mode
- [ ] PCI Read With Write Flush Mode
- [ ] Gate the Local Bus Latency Timer with BR
- [ ] PCI Read No Flush Mode
- [ ] Subsystem ID and Subsystem Vendor ID

Local Bus Latency: 1C

Local Bus Pause Timer: 00

DMA Channel Priority: [✓ Rotational priority scheme] Channel 0 has priority / Channel 1 has priority

Figure A-3 Direct Master Mode / Arbitration dialog box



Big/Little Endian Descriptor Register

- [✓] Configuration Register Big Endian Mode
- [ ] Direct Master Big Endian Mode
- [ ] Direct Slave Address Space 0 Big Endian Mode
- [ ] Direct Slave Address Exp ROM 0 Big Endian Mode
- [ ] Big Endian Byte Lane Mode
- [ ] Direct Slave Address Space 1 Big Endian Mode
- [✓] DMA Channel 1 Big Endian Mode
- [✓] DMA Channel 0 Big Endian Mode

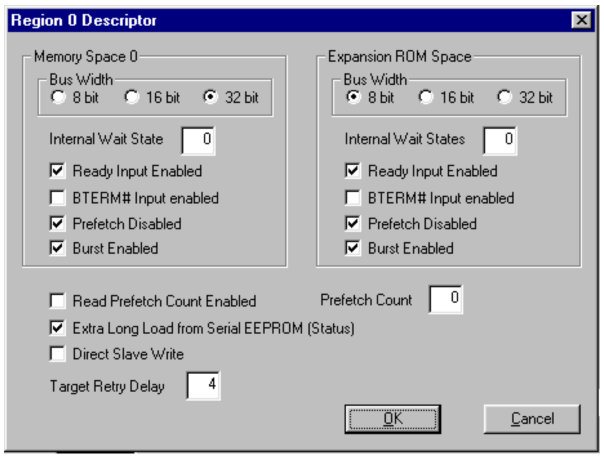
Figure A-4 Endian Descriptor dialog box

Endian Descriptor Dialog Box

The Endian Descriptor dialog box provides information on the current value of the Big/Little Endian Descriptor register and allows modification of that value (see Figure A-4).

The Local Space 0/Exp ROM Dialog Box

The Region 0 Descriptor dialog box provides information on the current value of the Local Address Space 0/Expansion ROM Bus Region Descriptor register and allows modification of that value (see Figure A-5).



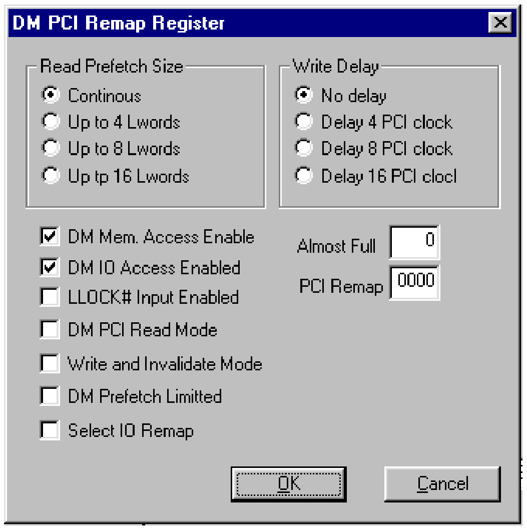
Region 0 Descriptor

Memory Space 0
Bus Width: ○ 8 bit ○ 16 bit ● 32 bit
Internal Wait State: 0
[✓] Ready Input Enabled
[ ] BTERM# Input enabled
[✓] Prefetch Disabled
[✓] Burst Enabled

Expansion ROM Space
Bus Width: ● 8 bit ○ 16 bit ○ 32 bit
Internal Wait States: 0
[✓] Ready Input Enabled
[ ] BTERM# Input enabled
[✓] Prefetch Disabled
[✓] Burst Enabled

[ ] Read Prefetch Count Enabled
Prefetch Count: 0
[✓] Extra Long Load from Serial EEPROM (Status)
[ ] Direct Slave Write
Target Retry Delay: 4

Figure A-5 Local Space 0/Exp ROM dialog box



DM PCI Remap Register

Read Prefetch Size
● Continous
○ Up to 4 Lwords
○ Up to 8 Lwords
○ Up tp 16 Lwords

Write Delay
● No delay
○ Delay 4 PCI clock
○ Delay 8 PCI clock
○ Delay 16 PCI clocl

[✓] DM Mem. Access Enable
[✓] DM IO Access Enabled
[ ] LLOCK# Input Enabled
[ ] DM PCI Read Mode
[ ] Write and Invalidate Mode
[ ] DM Prefetch Limitted
[ ] Select IO Remap

Almost Full: 0
PCI Remap: 0000

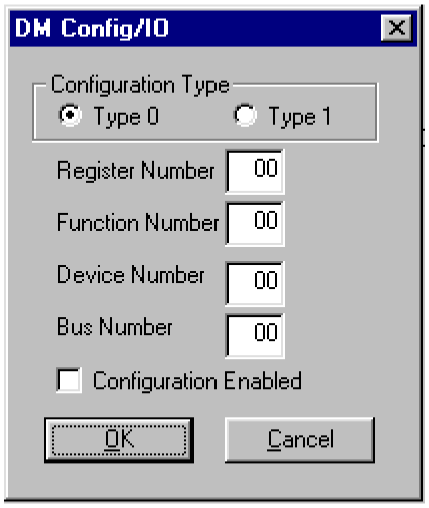
Figure A-6 Direct Master PCI Remap

The DM PCI Remap Dialog Box

The DM PCI Remap dialog box provides information on the current value of the PCI Base Address (Remap) Register for Direct Master to PCI Memory and allows modification of that value (see Figure A-6).

The DM Configuration I/O Address Dialog Box

The DM Configuration I/O Address dialog box provides information on the current value of the PCI configuration Address Register for Direct Master to PCI I/O-CFG and allows for modification of that value (see Figure A-7).


DM Config/IO

Configuration Type: [✓ Type 0] Type 1

Register Number: 00

Function Number: 00

Device Number: 00

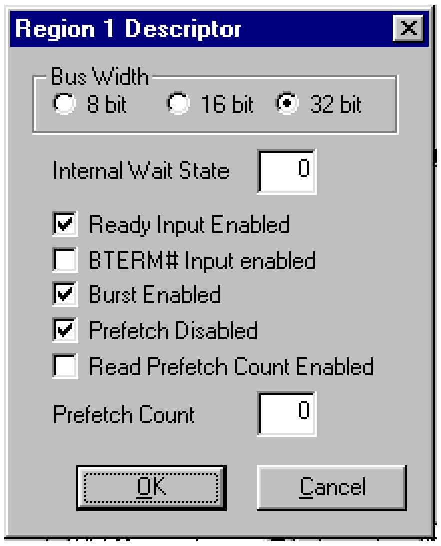
Bus Number: 00

☐ Configuration Enabled

Figure A-7 Direct Master Configuration I/O Address dialog box

The Region 1 Dialog Box

The Region 1 dialog box provides information on the Local Address Space 1 Bus Region Descriptor register and allows modification of that value (see Figure A-8).


Region 1 Descriptor

Bus Width: 8 bit 16 bit [✓ 32 bit]

Internal Wait State: 0

☑ Ready Input Enabled

☐ BTERM# Input enabled

☑ Burst Enabled

☑ Prefetch Disabled

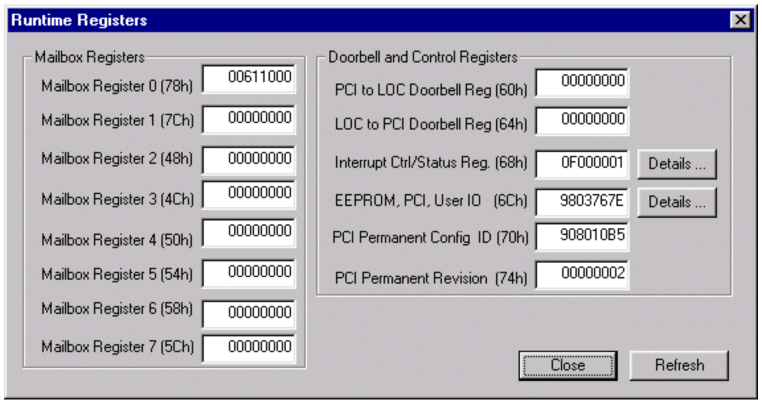
☐ Read Prefetch Count Enabled

Prefetch Count: 0

Figure A-8 Region 1 dialog box

A.1.3 The Runtime Register Group Dialog Box

The Runtime Register Group dialog box displays the current register values. All register values can be modified by changing the contents of any register.



Runtime Registers

Mailbox Registers

Mailbox Register 0 (78h)	00611000
Mailbox Register 1 (7Ch)	00000000
Mailbox Register 2 (48h)	00000000
Mailbox Register 3 (4Ch)	00000000
Mailbox Register 4 (50h)	00000000
Mailbox Register 5 (54h)	00000000
Mailbox Register 6 (58h)	00000000
Mailbox Register 7 (5Ch)	00000000

Doorbell and Control Registers

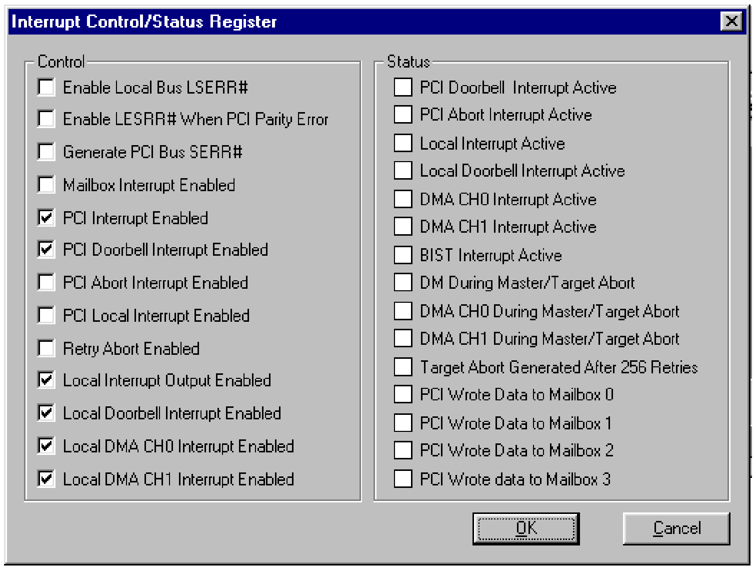
PCI to LOC Doorbell Reg (60h)	00000000
LOC to PCI Doorbell Reg (64h)	00000000
Interrupt Ctrl/Status Reg. (68h)	0F000001
EEPROM, PCI, User IO (6Ch)	9803767E
PCI Permanent Config ID (70h)	908010B5
PCI Permanent Revision (74h)	00000002

Figure A-9 Runtime Registers dialog box

The Interrupt Control/Status Register Dialog Box

The Interrupt Control/Status Register Dialog Box provides information on the current value of the Interrupt Control/Status register.

The information contained in the dialog box is grouped into two categories, the Control bits and the Status bits. The control bits enable triggering of interrupts for certain events, such as DMA events, doorbell events and others. The status bits cannot be modified directly. They show the current status of the various interrupt triggers.



Interrupt Control/Status Register

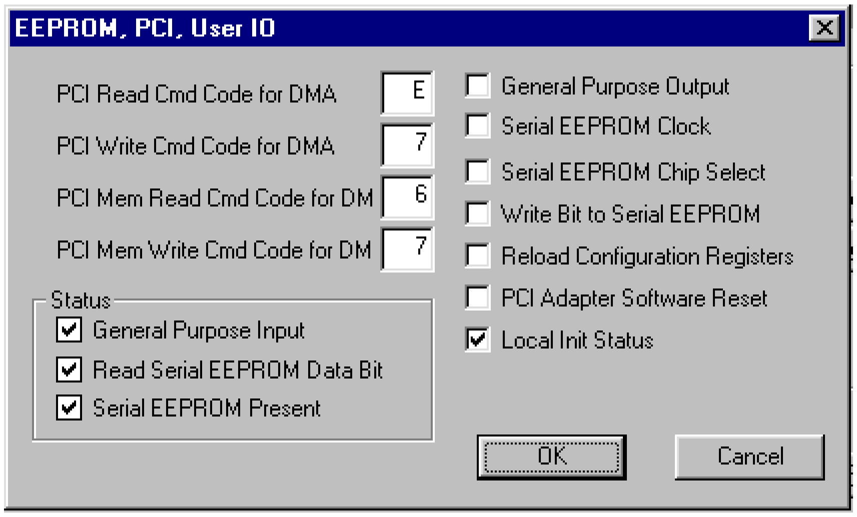
Control
- [ ] Enable Local Bus LSERR#
- [ ] Enable LESRR# When PCI Parity Error
- [ ] Generate PCI Bus SERR#
- [ ] Mailbox Interrupt Enabled
- [✓] PCI Interrupt Enabled
- [✓] PCI Doorbell Interrupt Enabled
- [ ] PCI Abort Interrupt Enabled
- [ ] PCI Local Interrupt Enabled
- [ ] Retry Abort Enabled
- [✓] Local Interrupt Output Enabled
- [✓] Local Doorbell Interrupt Enabled
- [✓] Local DMA CH0 Interrupt Enabled
- [✓] Local DMA CH1 Interrupt Enabled

Status
- [ ] PCI Doorbell Interrupt Active
- [ ] PCI Abort Interrupt Active
- [ ] Local Interrupt Active
- [ ] Local Doorbell Interrupt Active
- [ ] DMA CH0 Interrupt Active
- [ ] DMA CH1 Interrupt Active
- [ ] BIST Interrupt Active
- [ ] DM During Master/Target Abort
- [ ] DMA CH0 During Master/Target Abort
- [ ] DMA CH1 During Master/Target Abort
- [ ] Target Abort Generated After 256 Retries
- [ ] PCI Wrote Data to Mailbox 0
- [ ] PCI Wrote Data to Mailbox 1
- [ ] PCI Wrote Data to Mailbox 2
- [ ] PCI Wrote data to Mailbox 3

Figure A-10 Interrupt Control and Status dialog box

The EEPROM, PCI, User IO Dialog Box

The EEPROM, PCI, User IO dialog box provides information on the current contents of the EEPROM Control, PCI Command Codes, User I/O Control, Initialization Control Register and allows modification of that value (see Figure A-11). The Status section contained in this dialog box contains values that cannot be modified.



EEPROM, PCI, User IO

PCI Read Cmd Code for DMA	E
PCI Write Cmd Code for DMA	7
PCI Mem Read Cmd Code for DM	6
PCI Mem Write Cmd Code for DM	7

Status
- [✓] General Purpose Input
- [✓] Read Serial EEPROM Data Bit
- [✓] Serial EEPROM Present

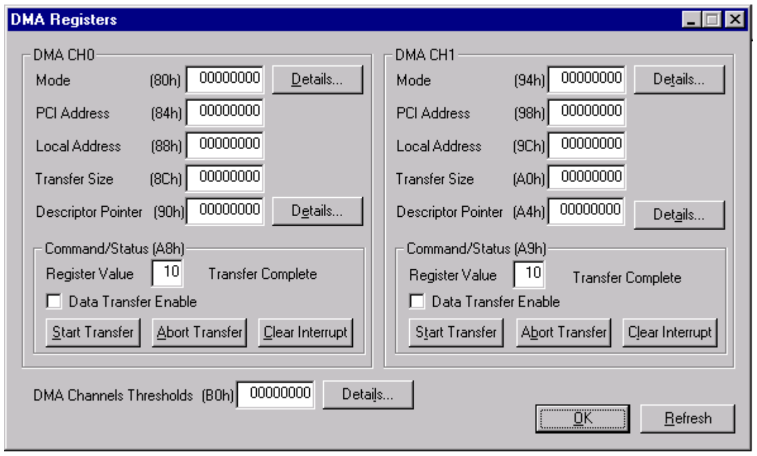
- [ ] General Purpose Output
- [ ] Serial EEPROM Clock
- [ ] Serial EEPROM Chip Select
- [ ] Write Bit to Serial EEPROM
- [ ] Reload Configuration Registers
- [ ] PCI Adapter Software Reset
- [✓] Local Init Status

Figure A-11 EEPROM, PCI, User IO Details dialog box

A.1.4 The DMA Register Group Dialog Box

The DMA Register Group dialog box contains the current values for the DMA registers for both DMA channels (see Figure A-12).

The `Start` and `Abort Transfer` buttons, in Figure A-12, initiate and terminate the DMA transfer using the current information provided in the DMA registers for the given DMA channel. The `Clear Interrupt` button resets the interrupts to their default state. The `Data Transfer Enable` bit, enables DMA transfers and activates the `Start`, `Abort`, and `Clear Interrupt` buttons.



DMA Registers

DMA CH0			DMA CH1		
Mode	(80h)	00000000	Mode	(94h)	00000000
PCI Address	(84h)	00000000	PCI Address	(98h)	00000000
Local Address	(88h)	00000000	Local Address	(9Ch)	00000000
Transfer Size	(8Ch)	00000000	Transfer Size	(A0h)	00000000
Descriptor Pointer	(90h)	00000000	Descriptor Pointer	(A4h)	00000000

Command/Status (A8h): Register Value 10, Transfer Complete, [ ] Data Transfer Enable

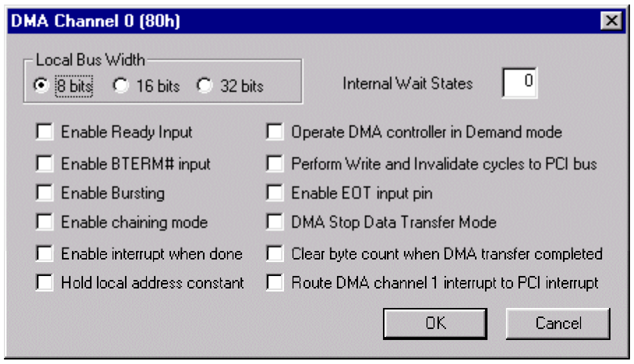
Command/Status (A9h): Register Value 10, Transfer Complete, [ ] Data Transfer Enable

DMA Channels Thresholds (B0h) 00000000

Figure A-12 DMA Registers dialog box

The DMA Mode Dialog Box

The DMA Mode dialog box provides information on the current value the DMA Channel's Mode Register and allows modification of that value (see Figure A-13).

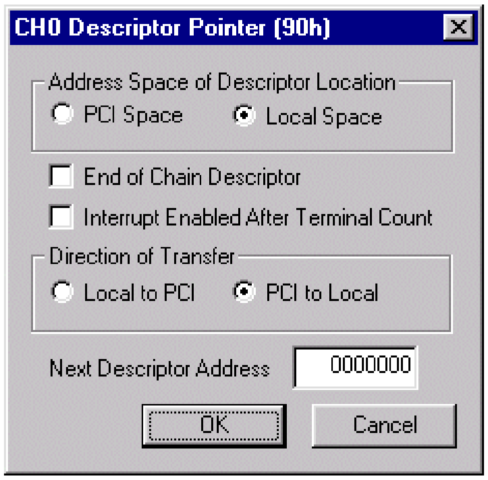

DMA Channel 0 (80h)

Local Bus Width: [✓ 8 bits] 16 bits 32 bits

Internal Wait States: 0

☐ Enable Ready Input
☐ Enable BTERM# input
☐ Enable Bursting
☐ Enable chaining mode
☐ Enable interrupt when done
☐ Hold local address constant
☐ Operate DMA controller in Demand mode
☐ Perform Write and Invalidate cycles to PCI bus
☐ Enable EOT input pin
☐ DMA Stop Data Transfer Mode
☐ Clear byte count when DMA transfer completed
☐ Route DMA channel 1 interrupt to PCI interrupt

Figure A-13 DMA Mode dialog box


CH0 Descriptor Pointer (90h)

Address Space of Descriptor Location: PCI Space [✓ Local Space]

☐ End of Chain Descriptor
☐ Interrupt Enabled After Terminal Count

Direction of Transfer: Local to PCI [✓ PCI to Local]

Next Descriptor Address: 0000000

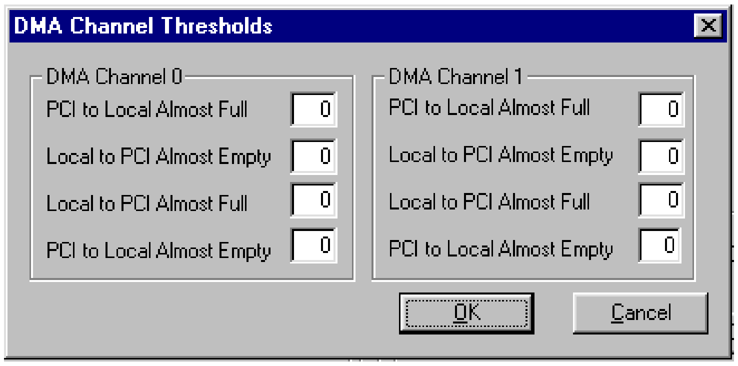
Figure A-14 Descriptor Pointer dialog box

The Descriptor Pointer Dialog Box

The Descriptor Pointer dialog box provides information on the current value of the DMA Channel's Descriptor Pointer Register and allows modification of that value (see Figure A-14).

The DMA Channels Threshold Dialog Box

The DMA Channels Threshold dialog box provides information on the current value of the DMA Threshold Register and allows modification of that value (see Figure A-15).

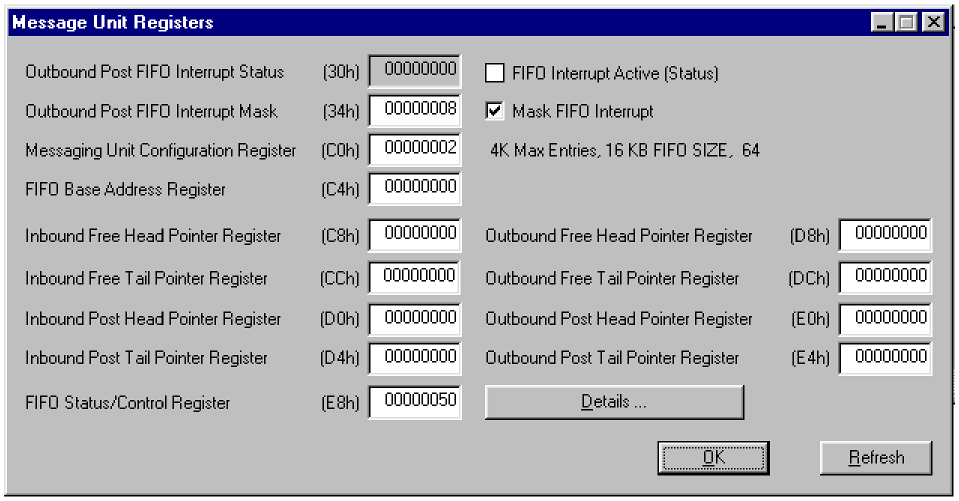

DMA Channel Thresholds

	DMA Channel 0	DMA Channel 1
PCI to Local Almost Full	0	0
Local to PCI Almost Empty	0	0
Local to PCI Almost Full	0	0
PCI to Local Almost Empty	0	0

Figure A-15 Threshold dialog box

A.1.5 The Messaging FIFO Register Group Dialog Box

The Messaging FIFO Register Group dialog box contains the current values for the Messaging FIFO Registers as shown in Figure A-16.



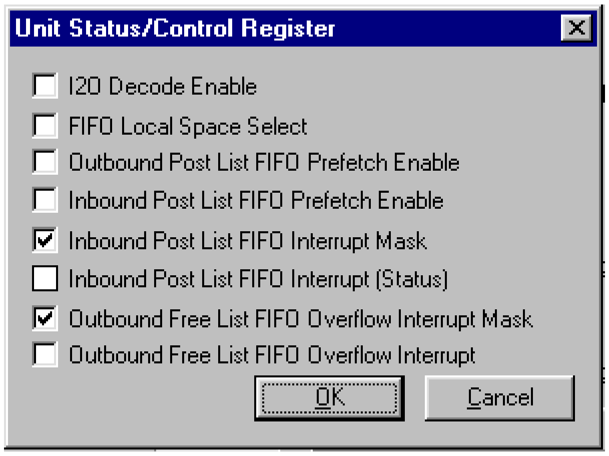
Outbound Post FIFO Interrupt Status	(30h)	00000000	☐ FIFO Interrupt Active (Status)		
Outbound Post FIFO Interrupt Mask	(34h)	00000008	☑ Mask FIFO Interrupt		
Messaging Unit Configuration Register	(C0h)	00000002	4K Max Entries, 16 KB FIFO SIZE, 64		
FIFO Base Address Register	(C4h)	00000000			
Inbound Free Head Pointer Register	(C8h)	00000000	Outbound Free Head Pointer Register	(D8h)	00000000
Inbound Free Tail Pointer Register	(CCh)	00000000	Outbound Free Tail Pointer Register	(DCh)	00000000
Inbound Post Head Pointer Register	(D0h)	00000000	Outbound Post Head Pointer Register	(E0h)	00000000
Inbound Post Tail Pointer Register	(D4h)	00000000	Outbound Post Tail Pointer Register	(E4h)	00000000
FIFO Status/Control Register	(E8h)	00000050			

Figure A-16 Messaging Unit Registers dialog box

All the register values can be modified with the exception of the Outbound Post FIFO Interrupt Status register. This register provides only the status of the Outbound Post FIFO interrupt and cannot be modified.

The FIFO Status/Control Register Dialog Box

The FIFO Status/Control Register dialog box provides information on the current value of the Queue Status/Control Register and allows modification of that value (see Figure A-17).



- ☐ I2O Decode Enable
- ☐ FIFO Local Space Select
- ☐ Outbound Post List FIFO Prefetch Enable
- ☐ Inbound Post List FIFO Prefetch Enable
- ☑ Inbound Post List FIFO Interrupt Mask
- ☐ Inbound Post List FIFO Interrupt (Status)
- ☑ Outbound Free List FIFO Overflow Interrupt Mask
- ☐ Outbound Free List FIFO Overflow Interrupt

Figure A-17 FIFO Status/Control Register dialog box

Appendix B. The PCI 9054 Register Set

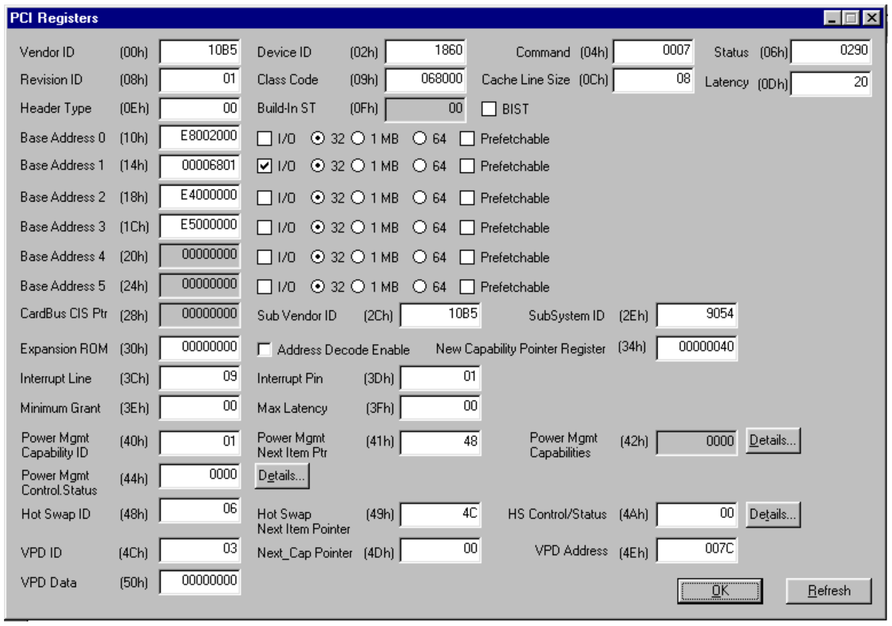
Each of the PCI 9054's register groups has a distinct dialog box. Each dialog box has the register values, the register's PCI based addresses, and a description of the register. Some registers have check boxes and radio buttons to help in describing and setting the register values. When required additional dialog boxes are available for more complex registers.

B.1 The Register Group Dialog Boxes

The PLXMon98's toolbar contains five buttons for register accesses. They are for `PCI Configuration Registers (PCR)`, `Local Configuration Registers (LCR)`, `Runtime Registers (RTR)`, `Local DMA Registers (LDR)`, and `Messaging Queue Registers (MQR)`.

B.1.1 PCI Configuration Register Group Dialog Box

The grayed text, in the `PCI Configuration Registers` dialog box indicates that the values cannot be modified using this dialog box. The radio buttons and check boxes, also in Figure B-1, indicate the current settings of the register bit fields. To update the contents of the dialog box push the `Refresh` button.

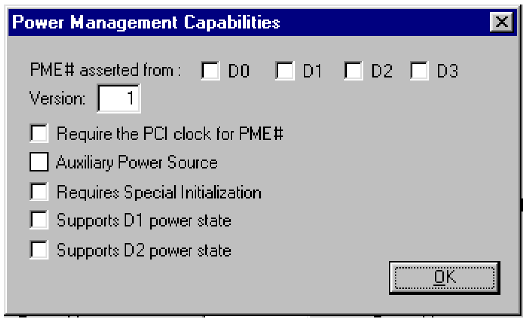


Vendor ID	(00h)	10B5	Device ID	(02h)	1860	Command	(04h)	0007	Status	(06h)	0290
Revision ID	(08h)	01	Class Code	(09h)	068000	Cache Line Size	(0Ch)	08	Latency	(0Dh)	20
Header Type	(0Eh)	00	Build-In ST	(0Fh)	00	☐ BIST					
Base Address 0	(10h)	E8002000	☐ I/O	[✓ 32] ○ 1 MB ○ 64	☐ Prefetchable						
Base Address 1	(14h)	00006801	☑ I/O	[✓ 32] ○ 1 MB ○ 64	☐ Prefetchable						
Base Address 2	(18h)	E4000000	☐ I/O	[✓ 32] ○ 1 MB ○ 64	☐ Prefetchable						
Base Address 3	(1Ch)	E5000000	☐ I/O	[✓ 32] ○ 1 MB ○ 64	☐ Prefetchable						
Base Address 4	(20h)	00000000	☐ I/O	[✓ 32] ○ 1 MB ○ 64	☐ Prefetchable						
Base Address 5	(24h)	00000000	☐ I/O	[✓ 32] ○ 1 MB ○ 64	☐ Prefetchable						
CardBus CIS Ptr	(28h)	00000000	Sub Vendor ID	(2Ch)	10B5	SubSystem ID	(2Eh)	9054			
Expansion ROM	(30h)	00000000	☐ Address Decode Enable			New Capability Pointer Register	(34h)	00000040			
Interrupt Line	(3Ch)	09	Interrupt Pin	(3Dh)	01						
Minimum Grant	(3Eh)	00	Max Latency	(3Fh)	00						
Power Mgmt Capability ID	(40h)	01	Power Mgmt Next Item Ptr	(41h)	48	Power Mgmt Capabilities	(42h)	0000	Details...		
Power Mgmt Control.Status	(44h)	0000	Details...								
Hot Swap ID	(48h)	06	Hot Swap Next Item Pointer	(49h)	4C	HS Control/Status	(4Ah)	00	Details...		
VPD ID	(4Ch)	03	Next_Cap Pointer	(4Dh)	00	VPD Address	(4Eh)	007C			
VPD Data	(50h)	00000000									

Figure B-1 PCI Configuration Registers dialog box

Power Management Capabilities

This dialog box seen in Figure B-2 displays Power Management setup attributes that are read only, or that can only be modified from the IOP side. If this dialog box is opened in Serial Mode, then certain values can be changed.


Power Management Capabilities

PME# asserted from : ☐ D0 ☐ D1 ☐ D2 ☐ D3

Version: 1

☐ Require the PCI clock for PME#

☐ Auxiliary Power Source

☐ Requires Special Initialization

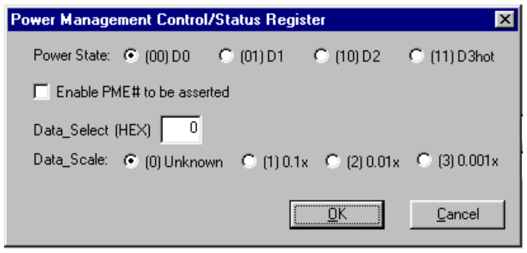
☐ Supports D1 power state

☐ Supports D2 power state

Figure B-2 Power Management Capabilities dialog box

Power Management Control/Status Register

In Figure B-3 are the bits that handle the operation of Power Management on the PCI 9054.


Power Management Control/Status Register

Power State: [✓ (00) D0] (01) D1 (10) D2 (11) D3hot

☐ Enable PME# to be asserted

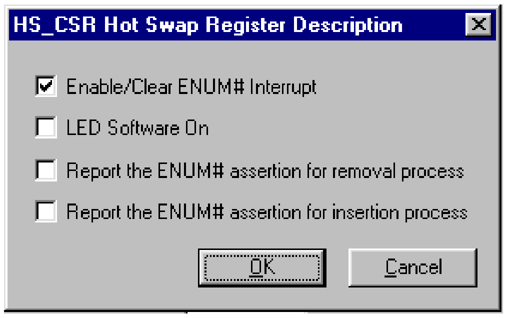
Data_Select (HEX) 0

Data_Scale: [✓ (0) Unknown] (1) 0.1x (2) 0.01x (3) 0.001x

Figure B-3 Power Management CSR dialog box

Hot Swap Control/Status Register

The Control and Status bits for Hot Swapping are found here in Figure B-4. All of these values can only be written from the PCI side.


HS_CSR Hot Swap Register Description

☑ Enable/Clear ENUM# Interrupt

☐ LED Software On

☐ Report the ENUM# assertion for removal process

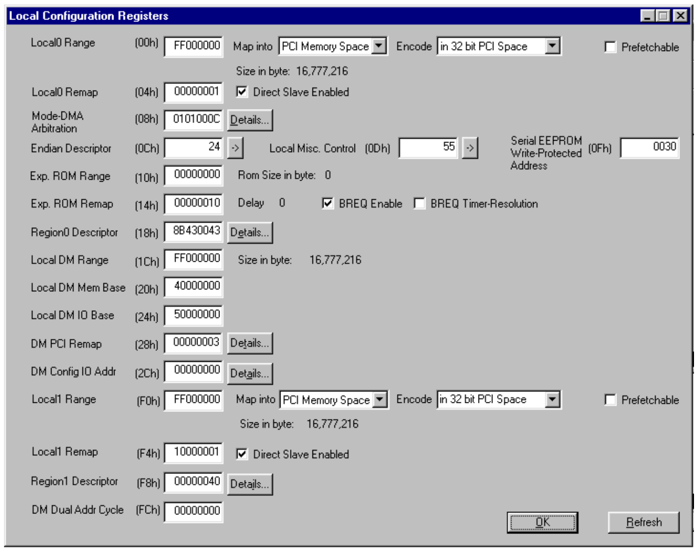
☐ Report the ENUM# assertion for insertion process

Figure B-4 Hot Swap CSR dialog box

B.1.2 Local Configuration Register Group Dialog Box

The Local Configuration register values are updated through the related edit and dialog boxes as seen in Figure B-5. The `size` text box reflects the value (in bytes) of the associated register. The memory size is calculated from corresponding register values and cannot be modified directly. To change the memory size, modify the associated register.

Seven registers within the Local Configuration Register Group have a more detailed dialog box and are as follows:



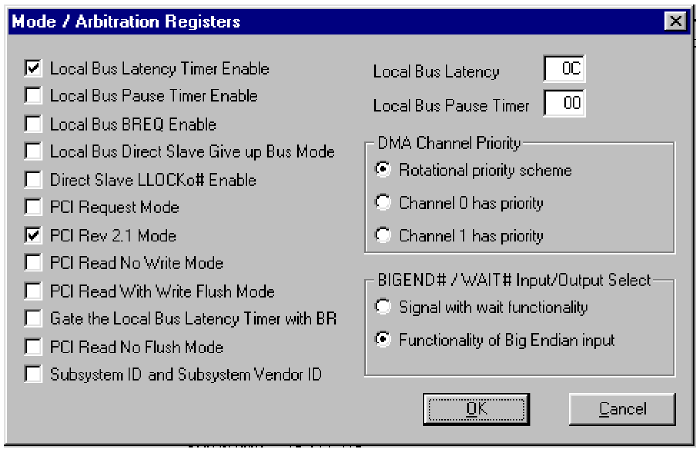
Local0 Range	(00h)	FF000000	Map into PCI Memory Space; Encode in 32 bit PCI Space; ☐ Prefetchable; Size in byte: 16,777,216
Local0 Remap	(04h)	00000001	☑ Direct Slave Enabled
Mode-DMA Arbitration	(08h)	0101000C	
Endian Descriptor	(0Ch)	24	
Local Misc. Control	(0Dh)	55	
Serial EEPROM Write-Protected Address	(0Fh)	0030	
Exp. ROM Range	(10h)	00000000	Rom Size in byte: 0
Exp. ROM Remap	(14h)	00000010	Delay 0; ☑ BREQ Enable; ☐ BREQ Timer-Resolution
Region0 Descriptor	(18h)	8B430043	
Local DM Range	(1Ch)	FF000000	Size in byte: 16,777,216
Local DM Mem Base	(20h)	40000000	
Local DM IO Base	(24h)	50000000	
DM PCI Remap	(28h)	00000003	
DM Config IO Addr	(2Ch)	00000000	
Local1 Range	(F0h)	FF000000	Map into PCI Memory Space; Encode in 32 bit PCI Space; ☐ Prefetchable; Size in byte: 16,777,216
Local1 Remap	(F4h)	10000001	☑ Direct Slave Enabled
Region1 Descriptor	(F8h)	00000040	
DM Dual Addr Cycle	(FCh)	00000000	

Figure B-5 Local Configuration Registers dialog box

- The Mode/Arbitration dialog box;
- The Endian Descriptor dialog box;
- The Miscellaneous Control Register dialog box;
- The Region 0/Exp ROM dialog box;
- The DM PCI Remap dialog box;
- The DM Config IO Address dialog box; and,
- The Region 1 dialog box.

The Mode/Arbitration Dialog Box

The Mode/Arbitration dialog box provides information on the current value Local/DMA Arbitration register and allows modification of that value (see Figure B-6).



Mode / Arbitration Registers

- [✓] Local Bus Latency Timer Enable
- [ ] Local Bus Pause Timer Enable
- [ ] Local Bus BREQ Enable
- [ ] Local Bus Direct Slave Give up Bus Mode
- [ ] Direct Slave LLOCKo# Enable
- [ ] PCI Request Mode
- [✓] PCI Rev 2.1 Mode
- [ ] PCI Read No Write Mode
- [ ] PCI Read With Write Flush Mode
- [ ] Gate the Local Bus Latency Timer with BR
- [ ] PCI Read No Flush Mode
- [ ] Subsystem ID and Subsystem Vendor ID

Local Bus Latency: 0C

Local Bus Pause Timer: 00

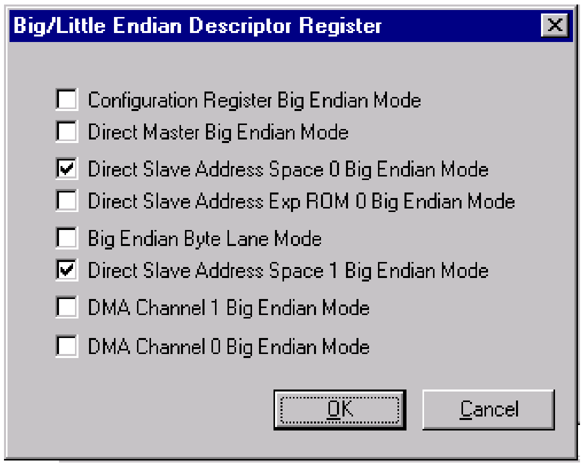
DMA Channel Priority
- [✓ Rotational priority scheme]
- Channel 0 has priority
- Channel 1 has priority

BIGEND# / WAIT# Input/Output Select
- Signal with wait functionality
- [✓ Functionality of Big Endian input]

Figure B-6 Mode/Arbitration dialog box

Endian Descriptor Dialog Box

The Endian Descriptor dialog box provides information on the current value of the Big/Little Endian Descriptor register and allows modification of that value (see Figure B-7).



Big/Little Endian Descriptor Register

- [ ] Configuration Register Big Endian Mode
- [ ] Direct Master Big Endian Mode
- [✓] Direct Slave Address Space 0 Big Endian Mode
- [ ] Direct Slave Address Exp ROM 0 Big Endian Mode
- [ ] Big Endian Byte Lane Mode
- [✓] Direct Slave Address Space 1 Big Endian Mode
- [ ] DMA Channel 1 Big Endian Mode
- [ ] DMA Channel 0 Big Endian Mode

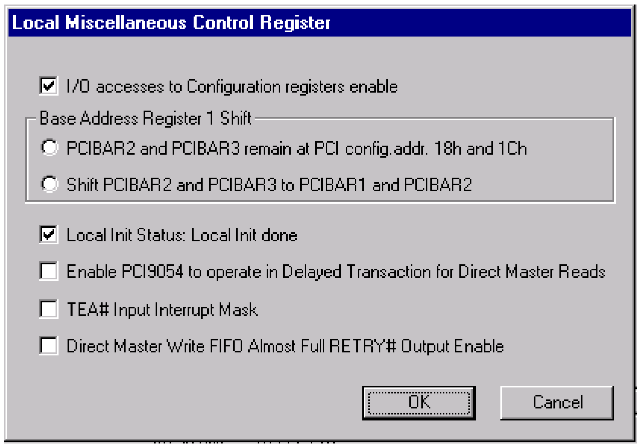
Figure B-7 Endian Descriptor Dialog Box

Local Miscellaneous Control Register Dialog Box

This dialog box contains bit controls for the miscellaneous functions of the PCI 9054 (see Figure B-8).

These functions include:

- Base Address Register 1 support.
- Init Done bit signal to BIOS.
- Direct Master Enables.
- Error interrupt Masks.



Local Miscellaneous Control Register

[✓] I/O accesses to Configuration registers enable

Base Address Register 1 Shift
- ( ) PCIBAR2 and PCIBAR3 remain at PCI config.addr. 18h and 1Ch
- ( ) Shift PCIBAR2 and PCIBAR3 to PCIBAR1 and PCIBAR2

[✓] Local Init Status: Local Init done

[ ] Enable PCI9054 to operate in Delayed Transaction for Direct Master Reads

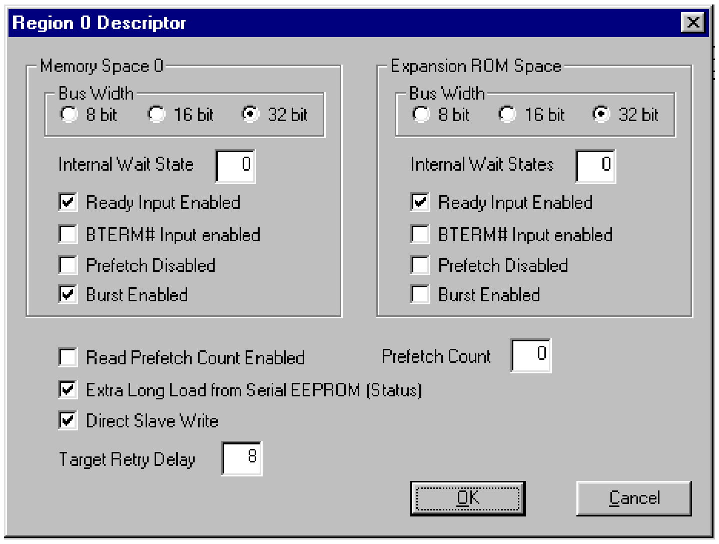
[ ] TEA# Input Interrupt Mask

[ ] Direct Master Write FIFO Almost Full RETRY# Output Enable

Figure B-8 Local Miscellaneous Control Register Dialog Box

The Local Space 0/Exp ROM dialog box

The Region 0 descriptor provides information on the current value of the Local Address Space 0/Expansion ROM Bus Region Descriptor register and allows modification of that value



Region 0 Descriptor

Memory Space 0
- Bus Width: ( ) 8 bit ( ) 16 bit (●) 32 bit
- Internal Wait State: 0
- [✓] Ready Input Enabled
- [ ] BTERM# Input enabled
- [ ] Prefetch Disabled
- [✓] Burst Enabled

Expansion ROM Space
- Bus Width: ( ) 8 bit ( ) 16 bit (●) 32 bit
- Internal Wait States: 0
- [✓] Ready Input Enabled
- [ ] BTERM# Input enabled
- [ ] Prefetch Disabled
- [ ] Burst Enabled

[ ] Read Prefetch Count Enabled — Prefetch Count: 0

[✓] Extra Long Load from Serial EEPROM (Status)

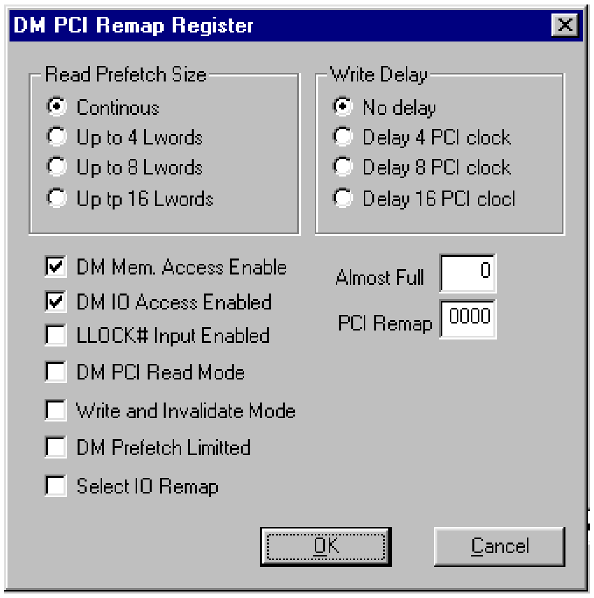
[✓] Direct Slave Write

Target Retry Delay: 8

Figure B-9 Region 0/ROM Descriptor dialog box

The DM PCI Remap Dialog Box

The DM PCI Remap dialog box provides information on the current value of the PCI Base Address (Remap) Register for Direct Master to PCI Memory and allows modification of that value (see Figure B-10).


DM PCI Remap Register

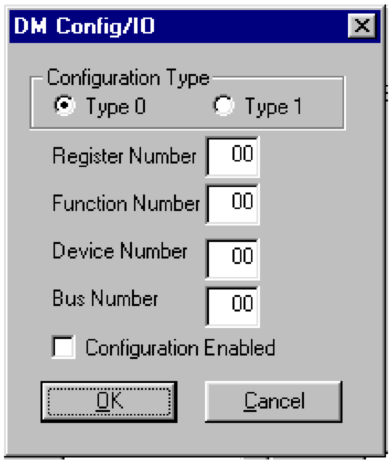
Read Prefetch Size
(●) Continous
( ) Up to 4 Lwords
( ) Up to 8 Lwords
( ) Up tp 16 Lwords

Write Delay
(●) No delay
( ) Delay 4 PCI clock
( ) Delay 8 PCI clock
( ) Delay 16 PCI clocl

[✓] DM Mem. Access Enable
[✓] DM IO Access Enabled
[ ] LLOCK# Input Enabled
[ ] DM PCI Read Mode
[ ] Write and Invalidate Mode
[ ] DM Prefetch Limitted
[ ] Select IO Remap

Almost Full: 0
PCI Remap: 0000

Figure B-10 Direct Master Remap dialog box


DM Config/IO

Configuration Type
(●) Type 0 ( ) Type 1

Register Number: 00
Function Number: 00
Device Number: 00
Bus Number: 00

[ ] Configuration Enabled

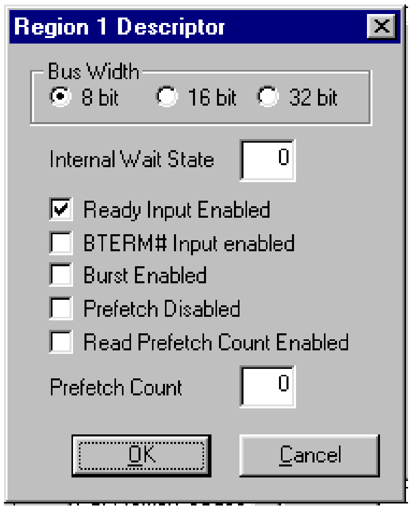
Figure B-11 Direct Master Config dialog box

The DM Configuration I/O Address dialog box

The DM Configuration I/O Address dialog box provides information on the current value of the PCI configuration Address Register for Direct Master to PCI I/O-CFG and allows modification of that value (see Figure B-11).

The Local Space 1 Dialog Box

The Region 1 dialog box provides information on the Local Address Space 1 Bus Region Descriptor register and allows modification of that value (see Figure B-12).



Region 1 Descriptor

Bus Width: [● 8 bit] ○ 16 bit ○ 32 bit

Internal Wait State: 0

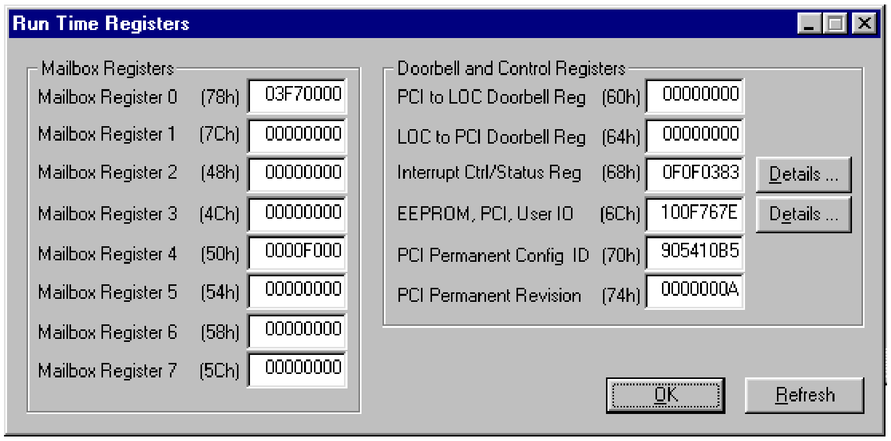
- [✓] Ready Input Enabled
- [ ] BTERM# Input enabled
- [ ] Burst Enabled
- [ ] Prefetch Disabled
- [ ] Read Prefetch Count Enabled

Prefetch Count: 0

Figure B-12 Region 1 Descriptor dialog box

B.1.3 The Runtime Register Group Dialog Box

The Run Time Register Group dialog box displays the current register values. All register values can be modified by changing the contents of any register.



Run Time Registers

Mailbox Registers

Mailbox Register 0	(78h)	03F70000
Mailbox Register 1	(7Ch)	00000000
Mailbox Register 2	(48h)	00000000
Mailbox Register 3	(4Ch)	00000000
Mailbox Register 4	(50h)	0000F000
Mailbox Register 5	(54h)	00000000
Mailbox Register 6	(58h)	00000000
Mailbox Register 7	(5Ch)	00000000

Doorbell and Control Registers

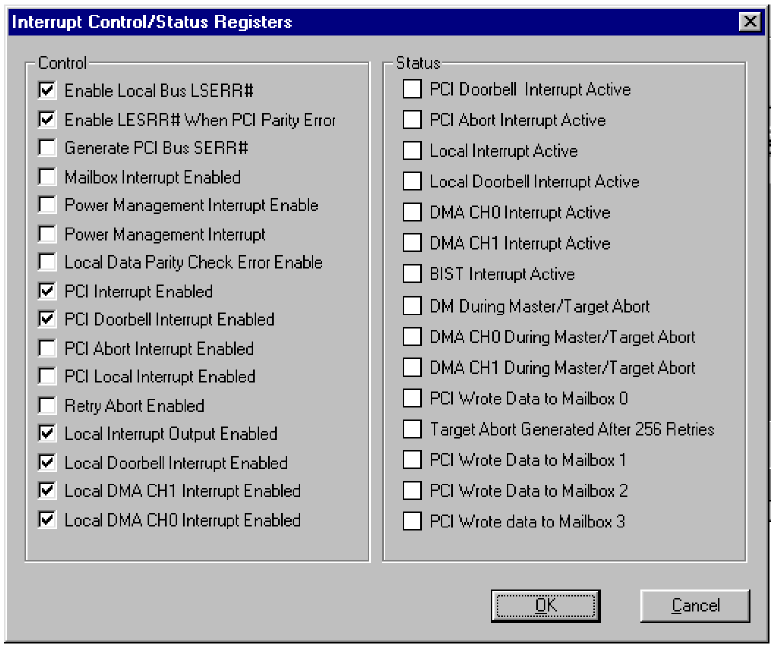
PCI to LOC Doorbell Reg	(60h)	00000000
LOC to PCI Doorbell Reg	(64h)	00000000
Interrupt Ctrl/Status Reg	(68h)	0F0F0383
EEPROM, PCI, User IO	(6Ch)	100F767E
PCI Permanent Config ID	(70h)	905410B5
PCI Permanent Revision	(74h)	0000000A

Figure B-13 Run Time Registers dialog box

The Interrupt Control/Status Register Dialog Box

The Interrupt Control/Status Register Dialog Box provides information on the current value of the Interrupt Control/Status register.

The information contained in the dialog box is grouped into two categories, the Control bits and the Status bits. The control bits enable triggering of interrupts for certain events, such as DMA events, doorbell events and others. The status bits cannot be modified directly. They show the current status of the various interrupt triggers.



Interrupt Control/Status Registers

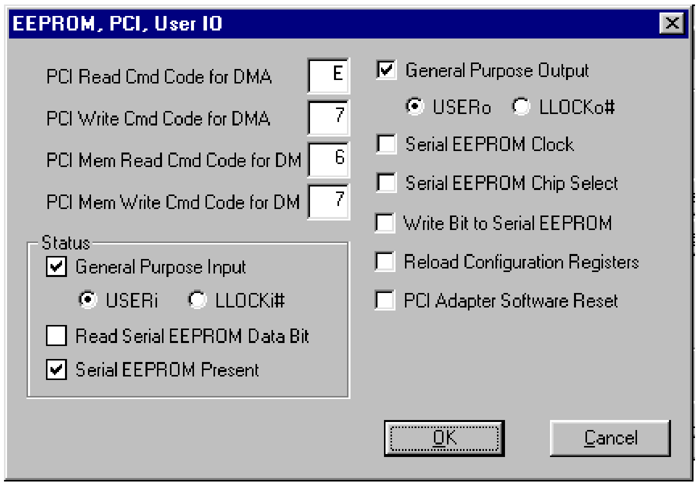
Control
- [✓] Enable Local Bus LSERR#
- [✓] Enable LESRR# When PCI Parity Error
- [ ] Generate PCI Bus SERR#
- [ ] Mailbox Interrupt Enabled
- [ ] Power Management Interrupt Enable
- [ ] Power Management Interrupt
- [ ] Local Data Parity Check Error Enable
- [✓] PCI Interrupt Enabled
- [✓] PCI Doorbell Interrupt Enabled
- [ ] PCI Abort Interrupt Enabled
- [ ] PCI Local Interrupt Enabled
- [ ] Retry Abort Enabled
- [✓] Local Interrupt Output Enabled
- [✓] Local Doorbell Interrupt Enabled
- [✓] Local DMA CH1 Interrupt Enabled
- [✓] Local DMA CH0 Interrupt Enabled

Status
- [ ] PCI Doorbell Interrupt Active
- [ ] PCI Abort Interrupt Active
- [ ] Local Interrupt Active
- [ ] Local Doorbell Interrupt Active
- [ ] DMA CH0 Interrupt Active
- [ ] DMA CH1 Interrupt Active
- [ ] BIST Interrupt Active
- [ ] DM During Master/Target Abort
- [ ] DMA CH0 During Master/Target Abort
- [ ] DMA CH1 During Master/Target Abort
- [ ] PCI Wrote Data to Mailbox 0
- [ ] Target Abort Generated After 256 Retries
- [ ] PCI Wrote Data to Mailbox 1
- [ ] PCI Wrote Data to Mailbox 2
- [ ] PCI Wrote data to Mailbox 3

Figure B-14 Interrupt Control and Status dialog box

The EEPROM, PCI, User IO Dialog Box

The EEPROM, PCI, User IO dialog box provides information on the current contents of the EEPROM Control, PCI Command Codes, User I/O Control, Init Control Register and allows modification of that value (see Figure B-15). The Status section contained in this dialog box contains values that cannot be modified.



EEPROM, PCI, User IO

PCI Read Cmd Code for DMA	E
PCI Write Cmd Code for DMA	7
PCI Mem Read Cmd Code for DM	6
PCI Mem Write Cmd Code for DM	7

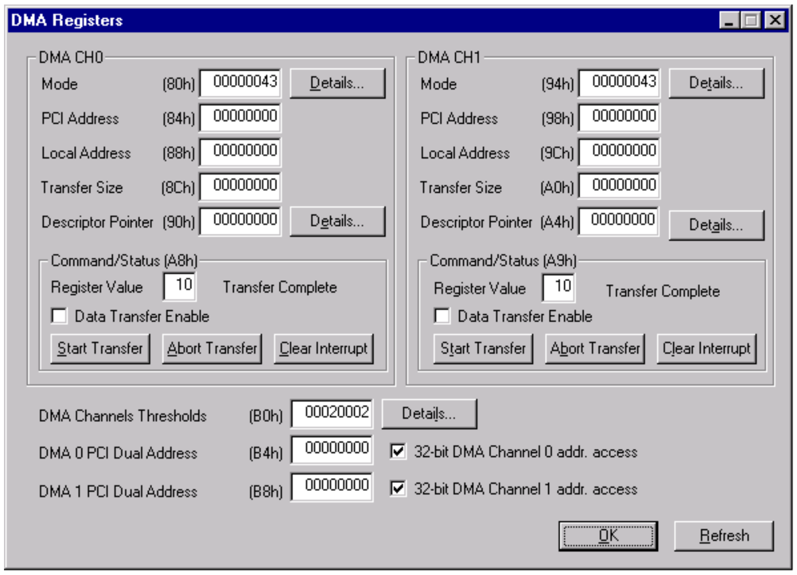
Status
- [✓] General Purpose Input
  - [✓ USERi] LLOCKi#
- [ ] Read Serial EEPROM Data Bit
- [✓] Serial EEPROM Present

- [✓] General Purpose Output
  - [✓ USERo] LLOCKo#
- [ ] Serial EEPROM Clock
- [ ] Serial EEPROM Chip Select
- [ ] Write Bit to Serial EEPROM
- [ ] Reload Configuration Registers
- [ ] PCI Adapter Software Reset

Figure B-15 EEPROM PCI User IO dialog box

B.1.4 The DMA Register Group Dialog Box

The DMA Register Group dialog box contains the current values for the DMA registers for both DMA channels (see Figure B-16).



DMA Registers

DMA CH0

Mode	(80h)	00000043	Details...
PCI Address	(84h)	00000000	
Local Address	(88h)	00000000	
Transfer Size	(8Ch)	00000000	
Descriptor Pointer	(90h)	00000000	Details...

Command/Status (A8h)
Register Value 10 Transfer Complete
[ ] Data Transfer Enable

DMA CH1

Mode	(94h)	00000043	Details...
PCI Address	(98h)	00000000	
Local Address	(9Ch)	00000000	
Transfer Size	(A0h)	00000000	
Descriptor Pointer	(A4h)	00000000	Details...

Command/Status (A9h)
Register Value 10 Transfer Complete
[ ] Data Transfer Enable

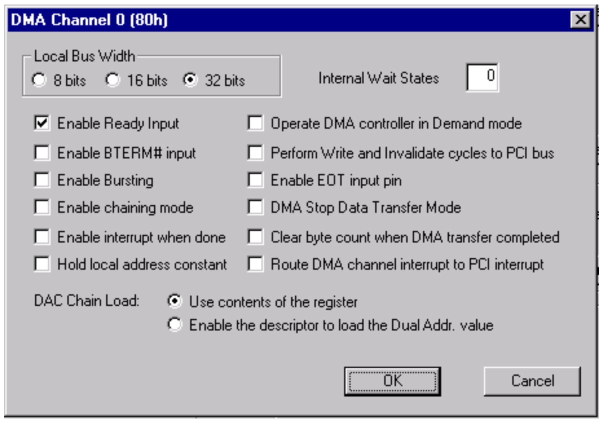
DMA Channels Thresholds	(B0h)	00020002	Details...
DMA 0 PCI Dual Address	(B4h)	00000000	[✓] 32-bit DMA Channel 0 addr. access
DMA 1 PCI Dual Address	(B8h)	00000000	[✓] 32-bit DMA Channel 1 addr. access

Figure B-16 Local DMA Registers dialog box

The `Start` and `Abort Transfer` buttons initiate and terminate the DMA transfer using the current information provided in the DMA registers for the given DMA channel. The `Clear Interrupt` button resets the interrupts to their default state. The `Data Transfer Enable` bit enables DMA transfers and activates the `Start`, `Abort`, and `Clear Interrupt` buttons.

The DMA Mode Dialog Box

The DMA Mode dialog box provides information on the current value the DMA Channel's Mode Register and allows modification of that value (see Figure B-17).



DMA Channel 0 (80h)

Local Bus Width: 8 bits 16 bits [✓ 32 bits]

Internal Wait States: 0

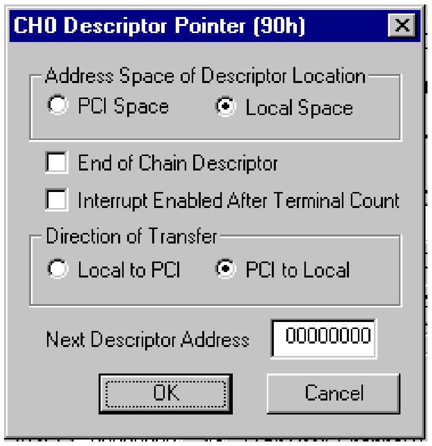
[✓] Enable Ready Input
[ ] Enable BTERM# input
[ ] Enable Bursting
[ ] Enable chaining mode
[ ] Enable interrupt when done
[ ] Hold local address constant
[ ] Operate DMA controller in Demand mode
[ ] Perform Write and Invalidate cycles to PCI bus
[ ] Enable EOT input pin
[ ] DMA Stop Data Transfer Mode
[ ] Clear byte count when DMA transfer completed
[ ] Route DMA channel interrupt to PCI interrupt

DAC Chain Load: [✓ Use contents of the register] Enable the descriptor to load the Dual Addr. value

Figure B-17 DMA Mode dialog box

The Descriptor Pointer Dialog Box

The Descriptor Pointer dialog box provides information on the current value of the DMA Channel’s Descriptor Pointer Register and allows modification of that value (see Figure B-18).


CH0 Descriptor Pointer (90h)

Address Space of Descriptor Location: PCI Space [✓ Local Space]

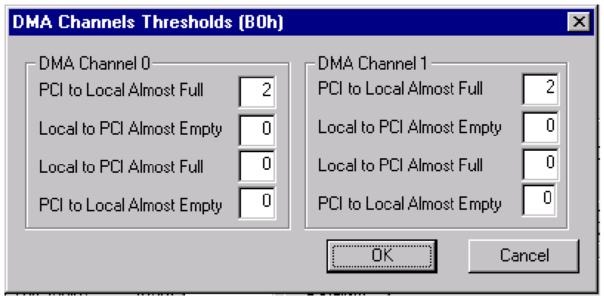
☐ End of Chain Descriptor

☐ Interrupt Enabled After Terminal Count

Direction of Transfer: Local to PCI [✓ PCI to Local]

Next Descriptor Address: 00000000

Figure B-18 DMA Descriptor Pointer dialog box


DMA Channels Thresholds (80h)

DMA Channel 0		DMA Channel 1	
PCI to Local Almost Full	2	PCI to Local Almost Full	2
Local to PCI Almost Empty	0	Local to PCI Almost Empty	0
Local to PCI Almost Full	0	Local to PCI Almost Full	0
PCI to Local Almost Empty	0	PCI to Local Almost Empty	0

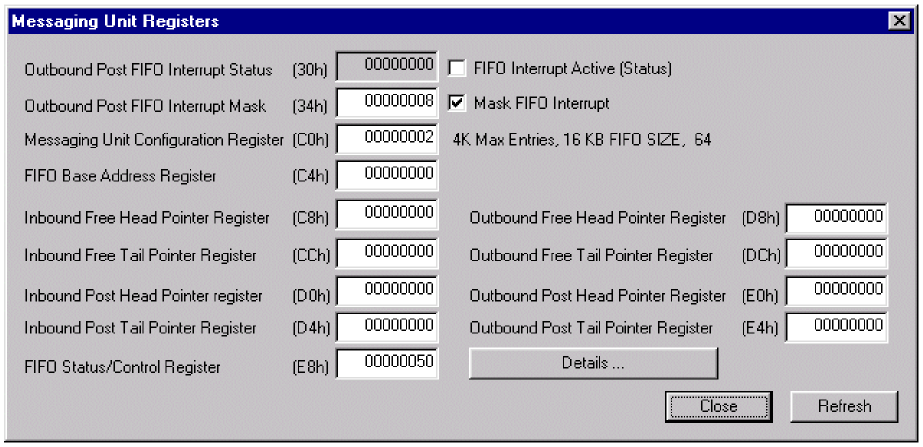
Figure B-19 DMA Thresholds dialog box

The DMA Channels Threshold Dialog Box

The DMA Channels Threshold dialog box provides information on the current value of the DMA Threshold Register and allows modification of that value (see Figure B-19).

B.1.5 The Messaging FIFO Register Group Dialog Box

The Messaging FIFO Register Group dialog box contains the current values for the Messaging FIFO Registers as shown in Figure B-20.



Messaging Unit Registers

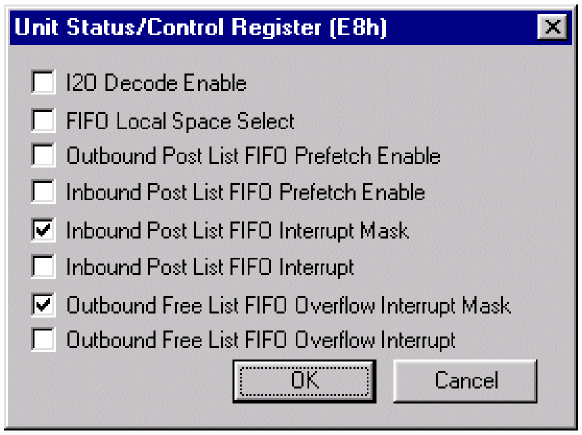
Outbound Post FIFO Interrupt Status	(30h)	00000000	☐ FIFO Interrupt Active (Status)		
Outbound Post FIFO Interrupt Mask	(34h)	00000008	☑ Mask FIFO Interrupt		
Messaging Unit Configuration Register	(C0h)	00000002	4K Max Entries, 16 KB FIFO SIZE, 64		
FIFO Base Address Register	(C4h)	00000000			
Inbound Free Head Pointer Register	(C8h)	00000000	Outbound Free Head Pointer Register	(D8h)	00000000
Inbound Free Tail Pointer Register	(CCh)	00000000	Outbound Free Tail Pointer Register	(DCh)	00000000
Inbound Post Head Pointer register	(D0h)	00000000	Outbound Post Head Pointer Register	(E0h)	00000000
Inbound Post Tail Pointer Register	(D4h)	00000000	Outbound Post Tail Pointer Register	(E4h)	00000000
FIFO Status/Control Register	(E8h)	00000050	Details ...		

Figure B-20 Messaging Unit Registers dialog box

All the register values can be modified with the exception of the Outbound Post FIFO Interrupt Status register. This register provides only the status of the Outbound Post FIFO interrupt and cannot be modified.

The FIFO Status/Control Register Dialog Box

The FIFO Status/Control Register dialog box provides information on the current value of the Queue Status/Control Register and allows modification of that value (see Figure B-21).



Unit Status/Control Register (E8h)

- ☐ I20 Decode Enable
- ☐ FIFO Local Space Select
- ☐ Outbound Post List FIFO Prefetch Enable
- ☐ Inbound Post List FIFO Prefetch Enable
- ☑ Inbound Post List FIFO Interrupt Mask
- ☐ Inbound Post List FIFO Interrupt
- ☑ Outbound Free List FIFO Overflow Interrupt Mask
- ☐ Outbound Free List FIFO Overflow Interrupt

Figure B-21 Status/Control Register dialog box

Appendix C. Troubleshooting and Customer Support

In this section one can find solutions to common problems found with PlxMon98. If a problem is encountered which is not listed here in the troubleshooting section, information is provided for customer support. Customer Support can be used to report bugs found as well.

I know I have an RDK in my computer, yet when I start PlxMon98, the program will only give me serial access. (WinNT only)

This means the driver was unable to find a “supported” device on your computer. When this happens, the driver will unload itself. Use the event viewer to verify this occurred and to check the cause. Use the driver wizard to add the vendor and device ID of your PCI device to the supported list. Then either restart the computer or manually restart the driver. Instructions for adding a supported device can be found in the SDK User’s Manual.

After installing my custom board (the Vendor and Device IDs are my own) the Add New Hardware Wizard in Windows 98 cannot find my board.

-or-

When adding two different engineering boards at the same time the Add New Hardware Wizard cannot differentiate between them. How do I know which board is which?

The “Add New Hardware Wizard” in Win98 relies on the Vendor and Device IDs of the PCI cards you are inserting. If a custom board is inserted, you must tell the Wizard that the .inf (installation script file) is located in the Inf directory under the Windows system directory. Then select “Unknown PCI XXXX board” depending on the PLX chip that is present on the RDK. The Inf directory is hidden, so make sure you “View all types” within the viewing options of Explorer to find it. Be sure to add new RDKs once at a time to avoid confusing the Wizard.

Appendix D. Glossary of Terms

Back End Monitor (BEM) or (BEM L1)

The Back End Monitor is an embedded program that can be compiled into the embedded software running on a PLX RDK. It's purpose is to scan the serial input and steal any data that it determines is a BEM command. The BEM commands allow for reads, writes, and resets of a PLX RDK. For more information about BEM, see the PLX SDK User's Manual.

COFF File Format

Coff files normally are the final data format for a RAM application compiled for use on the 9054RDK-860 board. The data contained within this file is big-endian.

IBM-401B Image File

Embedded programs compiled for the PCI 9080RDK-401B for RAM will be created in this format. The data is stored in big-endian format.

IC (Integrated Circuit)

While this term has many specific examples, this document uses this term to refer to the PLX IC (the PCI 9080 or PCI 9054 chip) exclusively.

IOP (Input/Output Platform)

This term is interchangeable with the Embedded platform or Local side. This can mean all the software and/or hardware that is on a PLX RDK.

Motorola SRecord

This file format is produced as an intermediate file when compiling code for the PCI 9054-RDK860. Data is not stored in any particular endian format.

PCI bus

The PCI bus physically is the location (along with a slot) where the PLX RDK is inserted. The PCI bus can also be given as an address range with data accessible according to the PCI specification.

PCI SDK 2.1

This is the current version of PLX's PCI Software Development Kit Version 2.1.

PlxMon98

An application which allows for use of PLX RDKs.

RDK (Reference Design Kit)

A PLX Reference Design Kit is the hardware for a specific board. Currently only two RDKs are supported for PCI SDK 2.1, the PCI 9080RDK-401B and the PCI 9054RDK-860